A Comparison of Schemas for Video Metadata Representation

Jane Hunter, CITEC, 317 Edward St Brisbane, Qld, 4001, Australia. Phone +617 33654310, Fax +617 33654311 jane@dstc.edu.au
Liz Armstrong, DSTC, Level 7, GP South, Uni of Qld, Qld, 4072, Australia. Phone +617 33654310, Fax +617 33654311 liz@dstc.edu.au

Abstract

To enable the resource discovery of audiovisual documents over the WWW, it will be necessary to define content description standards or metadata standards for complex, multi-layered, time-dependent information-rich audiovisual data streams. In particular, this is the primary goal of the emerging MPEG-7 standard, the "Multimedia Content Description Interface" [1], under development by the MPEG group. In the past, a lot of effort has gone into generating descriptors and description schemes for video indexing but comparatively little research has been done on schemas capable of defining the structure, content and semantics of video documents and enabling validation and higher levels of automated content checking. This paper compares the capabilities of the RDF Schema, Extensible Markup Language (XML) Document Type Definitions (DTD's), Document Content Description (DCD) and Schema for Object-Oriented XML (SOX), for supporting and validating hierarchical video descriptions based on Dublin Core, MPEG-7 and a specific hierarchical structure. Finally this paper proposes a hybrid schema based on features from each of these schemas which will satisfy the MPEG-7 Description Definition Language (DDL) requirements.

Keywords

Video, Metadata, Schema, Dublin Core, MPEG-7

1. Introduction

To enable the resource discovery of audiovisual documents over the WWW, it will be necessary to define content description standards or metadata standards for complex, multi-layered, time-dependent information-rich data streams. In particular, this is the primary goal of the developing MPEG-7 standard, the "Multimedia Content Description Interface" [1], under development by the MPEG group.

A number of papers have considered the application of Dublin Core (DC) and the Resource Description Framework (RDF) to video indexing [2, 3, 4, 5]. An example of such an application is described briefly below. However, very little work has been done on defining schemas which are capable of actually validating and constraining video descriptions and their associated data models. Such schemas will be necessary for the development of cost-efficient, user-friendly, semi-automatic metadata generation and editing tools for video. Such a schema would also provide a solution for the Description Definition Language (DDL) component of the MPEG7 requirements.

This paper first briefly presents a video description scheme based on Dublin Core and MPEG-7. From this description format, a list of schema requirements are generated. It then compares the ability of a number of existing schemas and schema proposals, including the RDF Schema, XML DTDs, DCD and SOX, to satisfy descriptions of hierarchical video structures. Examples of schema definitions are given to illustrate their capabilities.

Finally this paper proposes a hybrid schema based on specific features from each of these schemas and schema proposals which would satisfy the MPEG-7 Description Definition Language (DDL) requirements.

2. Proposed Video Description Scheme

Dublin Core was designed specifically for generating metadata to facilitate the resource discovery of textual documents. Although a number of workshops have been held to discuss the applicability of Dublin Core to non-textual documents such as images, sound and moving images, they have primarily focused on extensions to the 15 core elements through the use of subelements and schemes specific to audiovisual data, to describe bibliographic-type information rather than the actual content.

It has been shown [2] that it is possible to describe both the structure and fine-grained details of video content by using the fifteen Dublin Core elements plus qualifiers and encoding this within RDF . This "pure Dublin Core" approach provides multiple levels of descriptive information. At the top level the 15 basic Dublin Core elements can be used to describe the bibliographic type information about the complete document (e.g. Title, Author, Contributor, Date etc.). This enables non-specialist inter-disciplinary searching, independent of media type. Extensions or qualifiers to specific DC elements (Type, Description, Relation, Coverage) can be applied at the lower levels (scenes, shots, frames) to provide fine-grained, discipline- and media-specific searching (e.g. Description.Camera.Angle). The disadvantage of this approach is that the semantic refinement of Dublin Core through the use of

qualifiers eventually leads to a loss of semantic interoperability.

The alternative is a "hybrid" approach in which RDF (or some other framework) is used to combine both simple unqualified Dublin Core and MPEG-7 descriptors within a single description container. Dublin Core can be used for generic media-independent search and retrieval while MPEG-7 can be used for object-specific fine-grained queries. Our future research will compare and evaluate these two approaches for multimedia resource discovery and determine the best balance between semantic interoperability, extensibility and modularity. At this stage, we don't know the specific attributes of each level, we can only assume that each structural component will possess both a set of Dublin Core attributes plus a set of MPEG-7 attributes, as illustrated in Figure 1 below.

For example, if DC.Type = "Image.Moving.TV.News.Scene" then valid descriptors will include both the DC simple elements plus MPEG-7 descriptors such as script, transcript, editlist, keyframe etc. If DC.Type = "Image.Moving.TV.News.Scene.Shot" then valid descriptors will include both the DC elements plus keyframe, camera_distance, camera_angle, camera_motion, opening_transition, closing_transition. If DC.Type = "Image.Moving.TV.News.Scene.Shot.Frame" then a valid descriptors will be the DC elements plus colour_histogram.

Figure 1 shows the logical structure, the structural components and their associated Dublin Core attributes and some assumed MPEG-7 attributes for the proposed video description scheme.

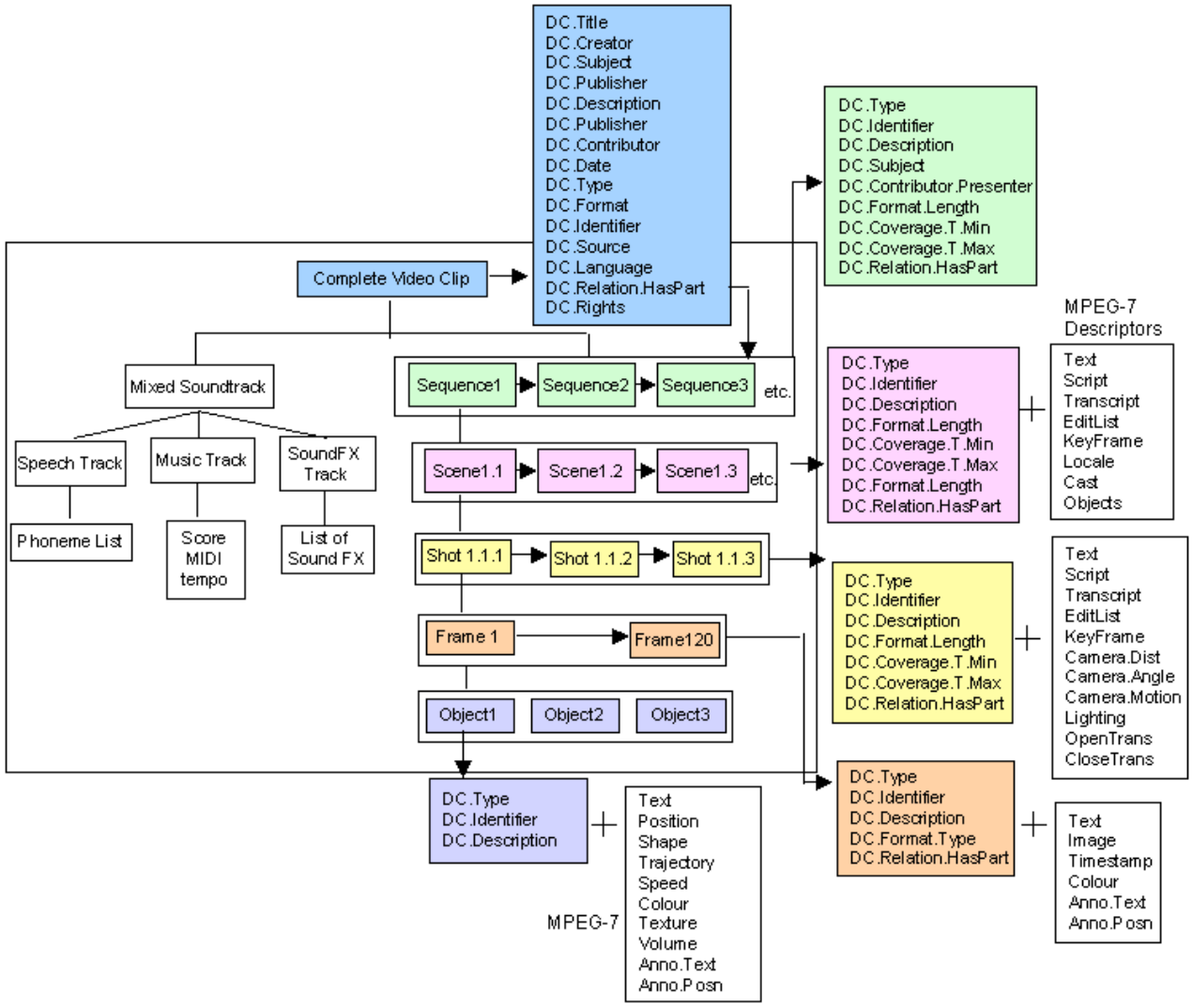

Figure 1: Multilayered Hierarchical Structure and Attributes of Video

---

3. Video Metadata Schema Requirements

In order to represent the video structure and Dublin Core descriptors outlined in Figure 1, a suitable schema must be able to support the following:

- Hierarchical structure definitions. The schema must be able to constrain the structure to a precise hierarchy in which complete video documents sit at the top level. These in turn contain sequences, which contain scenes, which contain shots, which contain frames, which contain objects or actors. Figure 1 illustrates this hierarchy.
- Each level (or class) within the hierarchy must be constrained to possess only specific attributes. In our description scheme, we assume that each layer possesses the 15 simple, optional and extensible DC elements plus a set of class-specific attributes

unique to that layer. These represent the set of MPEG-7 descriptors for that class when they become available.

- Element and attribute inheritance. It should be possible to specify sub-classing with inheritance of attributes and elements from the upper to lower classes. In addition, sub-classes should be able to have their own additional attributes and elements. This allows efficient reuse and customization of document schemas.
- Data Typing. It must be possible to constrain the values of attributes to certain data types. Data types supported should include primitive data types as well as Schemes (e.g. SMPTE), enumerated data types, controlled vocabularies, file types (images), URIs and complex data types (e.g.colour histograms, 3D vectors, graphs, RGB values etc.). It should also be possible to specify multiple alternative schemes or data types for a particular attribute.
- Cardinality within attributes should be representable. It must be possible to specify that an attribute can have zero, one or multiple values. Ideally the minimum and maximum number of attributes should also be specifiable e.g. a scene must contain between 2 and 5 shots.
- Spatio-temporal specifications. The Schema must be able to support the specification of temporal characteristics e.g. begin and end time of segments and their duration. Similarly, it should be able to support spatial representation e.g. regions within an image or motion along a line.
- Spatial, temporal and conceptual relations. Spatial relations such as neighbouring objects and temporal relations such as sequential or parallel segments should be supported. Given such a relationship between two classes, it should also be possible to constrain specific attribute values of these classes. For example, the start and end times of scenes contained within a sequence, must lie within the start and end time of that sequence.
- Human-readability. It is desirable rather than mandatory that both the schema and the description output from the schema should be human-readable.
- Availability of supporting technologies such as parsers (capable of validating input descriptions), databases and query languages.

These requirements are similar and compatible with the DDL requirements listed in section 4.1.1 of the MPEG-7 Requirements Document [7].

---

4. Resource Description Framework (RDF) Schema

The Resource Description Framework (RDF) enables interoperability between applications which exchange machine-understandable information on the Web. A model for representing metadata as well as a syntax for encoding RDF, based on XML has been defined in the RDF Model and Syntax Specification document [8].

RDF is based on a resource and property data model system. A collection of classes (typically authored for a specific purpose or domain) and the definition of their properties (attributes) and corresponding semantics represent an RDF schema. A schema defines not only the properties of the resource or class (Title, Author, Subject, Size, Color etc.) but also may define the kinds of resources being described (books, webpages, people, companies, etc.). The details of RDF schemas have been defined in the RDF Schema Specification document [9].

Classes are organized in a hierarchy, and offer extensibility through subclass refinement. This way, in order to create a schema slightly different from an existing one, one can just provide incremental modifications to the base schema. Through the sharability of schemas RDF will support the reusability of metadata definitions. Due to RDF's incremental extensibility, agents processing metadata will be able to trace the origins of schemes they are unfamiliar with back to known schemes, and perform meaningful actions on metadata they weren't originally designed to process. The sharability and extensibility of RDF also allows metadata authors to use multiple inheritance to "mix" definitions, to provide multiple views to their data, taking advantage of work done by others. The XML namespace mechanism serves to identify different RDF Schemas.

RDF schemas can be compared to XML Document Type Descriptions (DTDs). Unlike an XML DTD, which gives specific constraints on the syntactical structure of a document, an RDF schema provides semantical information about the interpretation of the statements given in an RDF data model. Given its goals, RDF appears to be the ideal approach for supporting descriptors from multiple description schemes simultaneously, as required by the MPEG-7 DDL.

4.1 Example of a Suitable RDF Schema

This section describes an RDF schema definition that attempts to map to the diagram in Figure 1 and support the requirements listed above.

Since we want the DC simple attributes to be applicable to every component or layer, *videos*, *sequences*, *scenes*, *shots*, *frames* and *objects* are all sub-classes of a top level *document* class which possesses the DC attributes. In addition each sub-class has its own additional descriptive properties or attributes which will correspond to MPEG-7 descriptors when they become available.

```
<rdf: RDF
         xmlns:rdf = "http://www.w3.org/TR/WD-rdf-syntax#"
         xmlns:rdfs= "http://www.w3.org/TR/WD-rdf-schema#"
         xmlns:dc= "http://purl.org/metadata/dublin_core#">

<rdfs:Class ID="MM_document">
<rdfs:comment>Class for representing a generic multimedia document</rdfs:comment>
```

```
</rdfs:Class>

<rdfs:comment>Define all of the DC elements for MM_document </rdfs:comment>

<rdf:PropertyType ID="Title">
<rdfs:comment>This is the DC Title element </rdfs:comment>
<rdfs:domain rdf:resource="#MM_document">
<rdfs:range rdf:resource="http://purl.org/metadata/dublin_core#Title"/>
</rdf:PropertyType>

<rdf:PropertyType ID="Creator">
<rdfs:comment>This is the DC Creator element </rdfs:comment>
<rdfs:domain rdf:resource="#MM_document">
<rdfs:range rdf:resource="http://purl.org/metadata/dublin_core#Creator"/>
</rdf:PropertyType>
.
etc.
.


<rdfs:Class ID="Video">
<rdfs:comment>Class for representing a video document. It is a subclass of MM_document</rdfs:comment>
<rdfs:subClassOf  rdf:resource="#MM_document"/>
</rdfs:Class>

<rdfs:Class ID="Sequence">
<rdfs:comment>Class for representing a sequence from a video document. It is a subclass of MM_document</rdfs:comment>
<rdfs:subClassOf  rdf:resource="#MM_document"/>
</rdfs:Class>

<rdfs:Class ID="Scene">
<rdfs:comment>Class for representing a scene. It is a subclass of MM_document</rdfs:comment>
<rdfs:subClassOf  rdf:resource="#MM_document"/>
</rdfs:Class>

<rdfs:Class ID="Shot">
<rdfs:comment> Class representing a shot</rdfs:comment>
<rdfs:subClassOf  rdf:resource="#MM_document"/>
</rdfs:Class>

<rdfs:Class ID="Frame">
<rdfs:comment> Represents a single frame. It is a subclass of #MM_document</rdfs:comment>
<rdfs:subClassOf rdf:resource="#MM_document"/>
</rdfs:Class>

<rdfs:Class ID="Object">
<rdfs:comment> Represents an object within a frame. It is a subclass of #MM_document</rdfs:comment>
<rdfs:subClassOf rdf:resource="#MM_document"/>
</rdfs:Class>
```

One of the problems with RDF is to create a generic property such as *contains* by which the hierarchical structure can be defined i.e. *videos* contain *sequences* which contain *shots* which contain *frames* which contain *objects* and *actors*. If you create a property *contains* for #video then how do you also apply it to #sequence, #scene and #shot? Since each property requires a single range, then generic relationships such as *contains* cannot be used. Instead, a separate property must be defined for each domain-range pair. This is tedious and repetitive. The lack of class-specific constraints on domain and range of properties is a major limitation of RDF, particularly when applied to complex multilayered documents in which you want to specify constraints on structural, spatial, temporal and conceptual relationships between components.

```
<rdf:PropertyType ID="contains_sequences">
<rdfs:comment> Property related to a video asset stating that a video consists of a number of sequences. </rdfs:comment>
<rdfs:domain rdf:resource="#Video">
<rdfs:range rdf: resource="#Sequence">
</rdfs:PropertyType>

<rdf:PropertyType ID="contains_scenes">
<rdfs:comment> Property related to a sequence asset stating that a sequnce consists of a number of scenes. </rdfs:comment>
<rdfs:domain rdf:resource="#Sequence">
<rdfs:range rdf: resource="#Scene">
</rdfs:PropertyType>

<rdf:PropertyType ID="contains_shots">
<rdfs:comment> Property related to a scene asset stating that a scene consists of a number of shots. </rdfs:comment>
<rdfs:domain rdf:resource="#Scene">
<rdfs:range rdf: resource="#Shot">
</rdfs:PropertyType>

<rdf:PropertyType ID="contains_frames">
<rdfs:comment> Property related to a shot asset stating that a shot consists of a number of frames. </rdfs:comment>
<rdfs:domain rdf:resource="#Shot">
<rdfs:range rdf: resource="#Frame">
</rdfs:PropertyType>

<rdf:PropertyType ID="contains_objects">
<rdfs:comment> Property related to a frame asset stating that a frame consists of a number of objects. </rdfs:comment>
<rdfs:domain rdf:resource="#Frame">
<rdfs:range rdf: resource="#Object">
</rdfs:PropertyType>
```

Another problem is the limited data typing within RDF. There are three ways of specifying data types within RDF:

- Use the primitive *Literal* data type available within the RDF schema definition. This is any quoted string.
- Implement a kind of enumerated data type by defining the range to be a class with a number of predefined instance values. This is used in the example below to define the possible values for shot transitions.
- Point to a separate namespace in which the data types have been defined. In the example below we refer to "http://www.w3.org/TR/datatypes" for any data types other than *literal*. This namespace doesn't currently exist but it is intended to define this within the W3C XML Schema Working Group [10] which has recently been set up.

Below is an example of the RDF Schema code defining some of the scene, shot , frame and object properties. It illustrates the three data typing methods available.

```
<rdf:PropertyType ID="startTime">
<rdfs:domain rdf:resource="#Scene">
<rdfs:domain rdf:resource="#Shot">
<rdfs:range rdf:resource="http://wwww.w3.org/TR/datatypes#Time"/>
</rdf:PropertyType>

<rdfs:PropertyType ID="keyFrame">
<rdfs:domain rdf:resource="#Scene">
<rdfs:domain rdf:resource="#Shot">
<rdfs:range rdf:resource="http://www.w3.org/TR/datatypes#Image"/>
</rdfs:PropertyType>

<rdfs:PropertyType ID="openTrans">
<rdfs:domain rdf:resource="#Shot">
<rdfs:range rdf:resource="#Transitions">
</rdfs:PropertyType>

<rdfs:PropertyType ID="closeTrans">
<rdfs:domain rdf:resource="#Shot">
<rdfs:range rdf:resource="#Transitions">
</rdfs:PropertyType>

<rdfs:Class ID="Transitions"/>
<Transitions ID="Cut"/>
<Transitions ID="Fade"/>
<Transitions ID="Wipe"/>
<Transitions ID="Dissolve"/>

<rdfs:PropertyType ID="position">
<rdfs:domain rdf:resource="#Object">
<rdfs:range rdf:resource="http://www.w3.org/TR/datatypes#Point">
</rdfs:PropertyType>

<rdfs:PropertyType ID="shape">
<rdfs:domain rdf:resource="#Object">
<rdfs:range rdf:resource="http://www.w3.org/TR/datatypes#Polygon">
</rdfs:PropertyType>

<rdfs:PropertyType ID="colorHistogram ">
<rdfs:domain rdf:resource="#Frame">
<rdfs:domain rdf:resource="#Object">
<rdfs:range rdf:resource="http://www.w3.org/TR/datatypes#Histogram">
</rdfs:PropertyType>
```

4.2 Advantages of the RDF Schema for Video Metadata

RDF Schemas, within the context of this application, have the following advantages:

- RDF Schema is able to provide meanings to elements or semantic structure not possible using purely syntactic schemas such as XML DTDs.. However the sorts of machine-understandable meanings provided in the current version of RDF Schema is very limited - so the advantage of "semantic validation" is virtually negligible.
- The other schemas really only provide implicit *child* or *contains* relationships between elements. With RDF you can specify any relationship types explicitly through properties but this is limited by the need to specify a single range. It isn't possible to constrain a particular relationship to multiple range/domain pairs e.g. sequences can only contains scenes which can only contain shots etc.
- Multiple namespaces. This enables the same feature to have different descriptors which correspond to different domains or description schemes. The ability to mix classification vocabularies within one XML-based encoding allows video authors or others to deliver richer domain-specific content descriptions thus increasing the re-usability of the video on the Web. This is a key requirement of the MPEG-7 DDL.
- Inheritance is supported through sub-classes and sub-properties. This provides easy extensibility and reuse of code.
- A simple RDF parser (SiRPAC [11]) exists but it has limited validation capabilities, checking only that the domain and range constraints are satisfied.

- It is human-readable, simple to understand and thus simple to extend or customize.

4.3 Limitations of the RDF Schema for Video Metadata

RDF Schema has the following problems or limitations:

- Unstable. The RDF Schema specifications are still under development and change frequently.
- Limited or no data typing. Almost all data typing will need to be provided by external namespaces, which don't yet exist.
- No cardinality. It isn't possible to specify optional, zero or multiple values for an attribute.
- Range constraints such as minimum and maximum values are not supported.
- Class-specific range constraints are not possible. Only one range is possible for a given property. The only way to do provide multiple ranges is to create multiple properties e.g. secs_start_time, frame_start_time, SMPTE_start_time.
- RDF Schema can't describe multilayered structures using a single generic "contains" property. This requires multiple specific "contains" properties i.e. "contains_sequences", "contains_scenes", "contains_shots", "contains_frames". The alternative is to implement code outside of the schema which understands DC.Relation.HasParts semantics and can perform the validation.
- Property-centricity makes readability difficult. The link between properties and classes is defined within the property definitions not the class definitions.
- No query language exists for RDF. Given a video structure, to find videos with similar structures, you need to be able to store RDF structures in a directed graph with associated attribute values in a database.
- The simplest way to specify spatial and temporal relationships is via the Collection elements: Seq, Bag and Alt, but these provide limited semantics. Since no <Par> element exists within RDF, the <Bag> element must be used to specify parallelism. For spatial relationships such as *neighbours*, if the list of neighbours is in a collection, can we assume that the first one is the nearest neighbour?
- Cannot map relationship-type properties between classes to constraints on the attribute values of the classes involved. For example, if two scenes abutt then their respective end and start frame numbers must be consecutive. If a sequence "contains" a scene, then the start and end times of the scene, must lie within the start and end times of the sequence. This is not supported by RDF Schema.
- RDF Schema is an incomplete mapping of the RDF Syntax and Data model. There are very useful features available within the RDF Syntax and Data Model Spec. which aren't supported in the RDF Schema.

---

5. XML DTDs

Extensible Markup Language (XML) Document Type Definitions (DTDs) provide a subset of SGML for describing documents. XML was developed by the XML Working Group under the World Wide Web Consortium (W3C) in 1996. The complete XML spec. is available from the W3C.[12].

Each XML document has both a logical and a physical structure. Physically, the document is composed of units called entities. An entity may refer to other entities to cause their inclusion in the document. A document begins in a "root" or document entity. Logically, the document is composed of declarations, elements, comments, character references, and processing instructions, all of which are indicated in the document by explicit markup. The logical and physical structures must nest properly.

The function of the markup in an XML document is to describe its storage and logical structure and to associate attribute-value pairs with its logical structures. XML provides the *document type declaration*, to define constraints on the logical structure and to support the use of predefined storage units. An XML document is valid if it has an associated document type declaration and if the document complies with the constraints expressed in it. Document type declarations are made in a Document Type Definition (DTD) file. The DTD file then contains a formal definition of a particular type of document outlining the element names and the structure of the document.

5.1 An Example of an XML DTD for Video Documents

The structure is defined in the *element* definitions at the top of the DTD. Each element has a set of associated attributes. All elements have an ID attribute plus the DC attributes. In addition, sequences, scenes and shots also have a set of time attributes (begin, end, duration). Each element also has its own set of level-specific attributes (which will correspond to the MPEG-7 descriptors when they become available).

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

<!DOCTYPE videodoc [

<!-- hierarchical structure of videodoc --!>
<!ELEMENT videodoc (sequence*) >
<!ELEMENT sequence (scene*)>
<!ELEMENT scene (shot*)>
<!ELEMENT shot (frame*)>
<!ELEMENT frame(object*)>
<!ELEMENT object(object*)>
```

```
<!-- ID attribute for every element --!>
<!ENTITY % id_attr "id ID #IMPLIED">


<!-- Set of Dublin Core Attributes --!>
<!ENTITY % dc_attr "
        Title    CDATA #IMPLIED
        Creator  CDATA #IMPLIED
        Subject  CDATA #IMPLIED
        Description CDATA #IMPLIED
        Publisher   CDATA #IMPLIED
        Contributor CDATA #IMPLIED
        Date        CDATA #IMPLIED
        Type        CDATA #IMPLIED
        Format      CDATA #IMPLIED
        Identifier  CDATA #IMPLIED
        Source      CDATA #IMPLIED
        Language    CDATA #IMPLIED
        Relation    CDATA #IMPLIED
        Coverage    CDATA #IMPLIED
        Rights      CDATA #IMPLIED">

<!ENTITY % scene_attr "
        Transcript   CDATA #IMPLIED
        Script       CDATA #IMPLIED
        EditList     CDATA #IMPLIED
        Keyframe     CDATA #IMPLIED
        Locale       CDATA #IMPLIED
        Cast         CDATA #IMPLIED
        Objects      CDATA #IMPLIED">

<!ENTITY % shot_attr "
        Keyframe     CDATA #IMPLIED
        CameraDist   NMTOKEN #IMPLIED
        CameraAngle  NMTOKEN #IMPLIED
        CameraMotion NMTOKEN #IMPLIED
        Lighting     NMTOKEN #IMPLIED
        OpenTrans    NMTOKEN #IMPLIED
        CloseTrans   NMTOKEN #IMPLIED">

<!ENTITY % frame_attr "
        Image              CDATA #IMPLIED
        Timestamp          CDATA #IMPLIED
        ColourText         NMTOKEN #IMPLIED
        ColourHistogram    CDATA #IMPLIED
        Texture            CDATA #IMPLIED
        Annotation         CDATA #IMPLIED
        Anno_Position      CDATA #IMPLIED">

<!ENTITY % object_attr "
        Position           CDATA #IMPLIED
        Shape              CDATA #IMPLIED
        Trajectory         CDATA #IMPLIED
        Speed              CDATA #IMPLIED
        ColourText         NMTOKEN  #IMPLIED
        ColourHistogram    CDATA #IMPLIED
        Texture            CDATA #IMPLIED
        Volume             CDATA #IMPLIED
        Annotation         CDATA #IMPLIED
        Anno_Position      CDATA #IMPLIED">

<!ENTITY % time_attr "
     begin CDATA #IMPLIED
     end   CDATA #IMPLIED
     dur   CDATA #IMPLIED">

<!ATTLIST videodoc
    %id_attr;
    %dc_attr;>

<!ATTLIST sequence
      %id_attr;
      %dc_attr;
      %time_attr;>

<!ATTLIST scene
      %id_attr;
      %dc_attr;
      %scene_attr;
      %time_attr;>

<!ATTLIST shot
      %id_attr;
      %dc_attr;
      %shot_attr;
      %time_attr;>

<!ATTLIST frame
      %id_attr;
      %dc_attr;
      %frame_attr;>
```

```
<!ATTLIST object
      %id_attr;
      %dc_attr;
      %object_attr;>

]>
```

5.2 Advantages of XML DTDs for Video Metadata

- Work is progressing on a query languages for XML e.g. XML-QL [13].
- XML parsers exist.
- Simplicity associated with a single namespace. Users only have to understand one namespace.
- XML is simpler than SGML, HyTime etc.
- XML DTDs are easy to read and understand. Short and sweet without all that data typing.
- Hierarchical structures are supported but only on a syntactical basis.

5.3 Disadvantages of XML DTDs for Video Metadata

- No name spaces. Since name spaces are not supported, definitions such as Dublin Core attributes will need to be redefined unless external entities are used. External entities provide a similar capability to namespaces. An external entity can be retrieved from an external DTD via a URI to this DTD and the entity's ID.
- Cardinality of attributes is zero or one in XML DTDs. This creates problems with DC attributes which are optional and repeatable. They may need to be declared as elements.
- There is very limited support for data typing. Only three kinds of attribute types are supported: a string type, a set of tokenized types and enumerated types. However Bray [14] has shown that it is possible to attach strong type declarations to XML elements using reserved attributes.
- It is a purely 'syntactic' machine-understandable schema which can't provide any of the semantics associated with complex structured multimedia data or support object-oriented data modelling concepts.
- There is no inheritance.
- There are no relationships possible other than the implicit *contains*.

---

6. Document Content Description (DCD) for XML

The Document Content Description (DCD) [15] facility for XML is an RDF vocabulary designed for describing constraints to be applied to the structure and content of XML documents. It consists of a set of properties used to constrain the types of elements and names of attributes that may appear in an XML document, the contents of the elements and the values of the attributes. It was designed to provide semantics over and above the purely syntactical XML DTDs. It was also designed to be conformant with the RDF Model and Syntax Specification (with some simplifications). DCD also incorporates a subset of an earlier submission to W3C, the XML-Data Submission [16].

The introduction to the XML-Data Submission says that it "describes an XML vocabulary for schemas, that is, for defining and documenting object classes. It can be used for classes which are strictly syntactic (for example, XML) or those which indicate concepts and relations among concepts (as used in relational databases, KR graphs and RDF). The former are called 'syntactic schemas;' the latter 'conceptual schemas'." Thus, XML-Data and DCD add object-oriented and data modelling concepts such as class inheritance to purely syntactic schemas such as XML DTDs.

DCD Schemas are based on *elements* and *attributes*. Elements correspond to RDF property types. DCD declarations constrain the content and attributes of elements in document instances, by assigning properties to objects of type *ElementDef* and *AttributeDef*.

6.1 Example of a DCD Schema

The DCD Schema below is based on the following assumptions:

- The Dublin Core elements are all described in a separate name space.
- The root element *video_doc* contains *video_sequences* which contains *video_scenes* etc.
- The Dublin Core elements apply to every level.
- In addition the sequence, scene and shot elements possess *start_time, end_time and duration* elements.
- In addition, each level has its own unique elements/attributes corresponding to MPEG-7 descriptors.

```
<DCD
   xmlns:DC="http://purl.org/metadata/dublin_core#"
   xmlns:CDT="http://www.w3.org/TR/complex_datatypes#">

   <?DCD syntax="explicit"?>
```

```
<Description>Example of a Video Document DCD</Description>
<Namespace>http://www.dstc.edu.au/schemas/videodcd</Namespace>

<ElementDef Type="videodoc" Model="Elements" Root="True">
   <Description>A video document structure.</Description>
   <Group RDF:Order="Seq">
      <Element>dc_values</Element>
      <Group Occurs="ZeroOrMore" RDF:Order="Seq">
          <Element>sequence</Element>
      </Group>
</ElementDef>

<ElementDef Type="sequence" Model="Elements">
   <Description>Description of a video sequence element</Description>
   <AttributeDef Name="seqID" Occurs="Required"/>
   <Group RDF:Order="Seq">
      <Element>dc_values</Element>
      <Element>time_attribs</Element>
      <Group Occurs="ZeroOrMore" RDF:Order="Seq">
          <Element>scene</Element>
      </Group>
   </Group>
</ElementDef>

<ElementDef Type="scene" Model="Elements">
   <Description>Description of a video scene element</Description>
   <AttributeDef Name="sceneID" Occurs="Required"/>
   <Group RDF:Order="Seq">
      <Element>dc_values</Element>
      <Element>time_attribs</Element>
      <Element>transcript</Element>
      <Element>keyframe</Element>
      <Group Occurs="ZeroOrMore" RDF:Order="Seq">
          <Element>shot</Element>
      </Group>
   </Group>
</ElementDef>

<ElementDef Type="shot" Model="Elements">
   <Description>Description of a video shot element</Description>
   <AttributeDef Name="shotID" Occurs="Required"/>
   <Group RDF:Order="Seq">
      <Element>dc_values</Element>
      <Element>time_attribs</Element>
      <Element>camera_distance</Element>
      <Element>camera_angle</Element>
      <Element>camera_motion</Element>
      <Element>lighting</Element>
      <Element>open_transition</Element>
      <Element>close_transition</Element>
      <Group Occurs="ZeroOrMore" RDF:Order="Seq">
          <Element>frame</Element>
      </Group>
   </Group>
</ElementDef>

<ElementDef Type="frame" Model="Elements">
   <Description>Description of a video frame element</Description>
   <AttributeDef Name="frameID" Occurs="Required"/>
   <Group RDF:Order="Seq">
      <Element>dc_values</Element>
      <Element>timestamp</Element>
      <Element>CDT:colourhistogram</Element>
      <Element>CDT:texture</Element>
      <Element>annotation</Element>
      <Element>CDT:anno_position</Element>
      <Group Occurs="ZeroOrMore" RDF:Order="Seq">
          <Element>object</Element>
      </Group>
   </Group>
</ElementDef>

<ElementDef Type="object" Model="Elements">
   <Description>Description of a video object/actor element</Description>
   <AttributeDef Name="objectID" Occurs="Required"/>
   <Group RDF:Order="Seq">
      <Element>dc_values</Element>
      <Element>CDT:position</Element>
      <Element>CDT:shape</Element>
      <Element>CDT:colourhistogram</Element>
      <Element>CDT:texture</Element>
      <Element>CDT:trajectory</Element>
      <Element>annotation</Element>
      <Element>CDT:anno_position</Element>
      <Group Occurs="ZeroOrMore" RDF:Order="Seq">
          <Element>object</Element>
      </Group>
   </Group>
</ElementDef>

<ElementDef Type="dc_values" Model="Elements">
   <Description>List of Dublin Core Elements</Description>
   <Group RDF:Order="Seq">
```

```
        <Element>DC:Title</Element>
        <Element>DC:Creator</Element>
        <Element>DC:Subject</Element>
        ......
      </Group>
   </ElementDef>

   <ElementDef Type="time_attribs" Model="Elements">
      <Group RDF:Order="Seq">
        <Element>start_time</Element>
        <Element>end_time</Element>
        <Element>duration</Element>
        ......
      </Group>
   </ElementDef>

   <ElementDef Type="transcript" Model="Data" Datatype="string">
   </ElementDef>
   <ElementDef Type="keyframe" Model="Data" Datatype="uri">
   </ElementDef>

   <ElementDef Type="camera_distance" Model="Data" Datatype="enumeration">
      <Values>close-up medium-shot long-shot</Values>
   </ElementDef>

   <ElementDef Type="camera_angle" Model="Data" Datatype="enumeration">
      <Values>low eye-level high</Values>
   </ElementDef>

   <ElementDef Type="open_transition" Model="Data" Datatype="enumeration">
      <Values>cut fade wipe dissolve</Values>
   </ElementDef>

   <ElementDef Type="annotation" Model="Data" Datatype="string">
   </ElementDef>


</DCD>
```

6.2 Advantages of DCD for Video Metadata

- Human-readable and simple.
- Provides better data typing than RDF Schemas and XML DTDs (but still only basic). Also provides upper and lower bound constraints on attribute values.
- Provides cardinality.
- Supports multiple namespaces.
- As an RDF vocabulary, it inherits the advantages of the data modelling concepts in RDF, plus constructs such as RDF:Seq and RDF:Alt.

6.3 Disadvantages of DCD for Video Metadata

- Currently no subclassing or inheritance but this is planned for the future. The proposal is to create subclasses from existing elements through an *extends* property.
- Only basic data typing is supported , not complex data types. There is no support for multiple alternate data types, except if you create alternate elements with different data types e.g. start_time value can be SMPTE, secs, frames(int). Also there is no support for constraining the values of certain attributes of related elements.
- Doesn't support data types such as points, lines, polygons, colour histograms etc. These would all have to be described in a separate namespace e.g."http://www.w3.org/TR/complex_datatypes". It is not possible to specify that just the element's datatype is to be a value from another namespace. You need to specify that the element itself is totally described in another namespace.
- Only *Seq* and or *Alt* Groups are available. *Bag* is not a legal value for the RDF:Order property. *Seq* is fine for specifying sequential components but for multimedia, there is also a need to support groups of elements which run in parallel. The RDF *Bag* element is the most suitable for specifying this, (in the absence of any *Par* value), but it isn't supported in DCD.

---

7. Schema for Object-Oriented XML (SOX)

Schema for Object-Oriented XML (SOX) [17] provides a facility for defining the structure, content and semantics of XML documents to enable XML validation and automated content checking.

SOX provides an alternative to XML DTDs for modeling markup relationships. The introduction to the SOX specification says that it provides the following advantages over XML DTDs:

- More efficient software development processes for distributed applications;

- Basic intrinsic datatypes;
- An extensible datatyping mechanism;
- Content model and attribute interface inheritance;
- A powerful namespace mechanism;
- Embedded documentation.

SOX supports three varieties of datatypes: *scalar datatypes*, *enumerated datatypes* and *format datatypes*. Scalar datatypes are derived from the basic *number* datatype, and support specification of the number of digits and decimal places, minimum and maximum value range, and a mask. An enumerated datatype may be derived from any of the intrinsic datatypes, and may specify an enumeration of valid values. A format datatype may be derived from any of the intrinsic datatypes, and must specify a mask.

In SOX, element types may inherit their content models and attribute definitions directly from another named element type. An element type may also inherit and extend an attribute list. Specialization of attribute definitions allows refinement and restriction of attribute datatype, enumeration list and default value. Additionally, an attribute value may be defined to be inherited from the identically named attribute on a parent or older ancestor element. Thus, for example, namespaces can be inherited from superordinate elements.

The SOX namespace facility enables *Objects* from any identifiable namespace to be used in building a SOX document. That is, any element, attribute, datatype, enumeration, entity, interface, notation, parameter, or processing instruction may be imported from any namespace.

A SOX document is a valid XML document, according to the SOX DTD. The schema designer is free to employ the same XML tools used for traditional XML documents. This means that a SOX document can processed by a validating XML parser, formatted according to an XSL stylesheet, and managed by any DOM-compliant or SAX-compliant application.

7.1 SOX Example

In this example, the structural elements, video_doc, video_sequence, video_scene, video_shot, video_frame and video_object are declared first. They each possess the DC attributes, plus their own specific elements and attributes.

```
<schema name="video_doc" namespace="http://www.dstc.edu.au/schemas/video_doc.xml" >

<h1>Video Metadata Document</h1>

<h2>Imported namespaces</h2>

<namespace name="dc" namespace="http://purl.org/metadata/dublin_core#"/>
<namespace name="dcq" namespace="http://purl.org/metadata/dublin_core_qualifiers#"/>

<h2>Structural Elements</h2>

<elementtype name="video_doc">
    <model>
       <sequence>
              <element name="dc_attributes"/>
              <element name="video_sequence" occurs="*"/>
      </sequence>
   </model>
</elementtype>

<elementtype name="video_sequence">
    <model>
       <sequence>
              <element name="seqID"/>
              <element name="dc_attributes"/>
              <element name="time_attributes"/>
              <element name="video_scene" occurs="*"/>
      </sequence>
   </model>
</elementtype>

<elementtype name="video_scene">
    <model>
       <sequence>
              <element name="sceneID"/>
              <element name="dc_attributes"/>
              <element name="time_attributes"/>
              <element name="transcript"/>
              <element name="key_frame"/>
              <element name="video_shot" occurs="*"/>
      </sequence>
   </model>
</elementtype>

<elementtype name="video_shot">
    <model>
       <sequence>
              <element name="shotID"/>
              <element name="dc_attributes"/>
```

```
              <element name="time_attributes"/>
              <element name="camera_distance"/>
              <element name="camera_angle"/>
              <element name="camera_motion"/>
              <element name="lighting"/>
              <element name="open_trans"/>
              <element name="close_trans"/>
              <element name="video_frame" occurs="*"/>
      </sequence>
   </model>
</elementtype>

<elementtype name="video_frame">
    <model>
       <sequence>
              <element name="frameID"/>
              <element name="dc_attributes"/>
              <element name="timestamp"/>
              <element name="colour_histogram"/>
              <element name="texture"/>
              <element name="video_object" occurs="*"/>
      </sequence>
   </model>
</elementtype>

<elementtype name="video_object">
    <model>
       <sequence>
              <element name="objectID"/>
              <element name="dc_attributes"/>
              <element name="position"/>
              <element name="shape"/>
              <element name="colour"/>
              <element name="texture"/>
              <element name="anno_text"/>
              <element name="anno_posn"/>
              <element name="video_object" occurs="*"/>
      </sequence>
   </model>
</elementtype>
```

The next step is to break down the elements to sub-elements and eventually data types. SOX supports both intrinsic basic datatypes as well as user-defined scalar, enumeration and formatted datatypes, derived from the intrinsic datatypes. The code below illustrates some of the capabilities of SOX data typing for video description.

```
<h2>Attribute Elements</h2>

<elementtype name="dc_attributes">
  <model>
       <sequence>
              <element namespace="dc" name="Title"/>
              <element namespace="dc" name="Creator"/>
              <element namespace="dc" name="Subject"/>
              .....
      </sequence>
  </model>
</elementtype>

<elementtype name="time_attributes">
  <model>
       <sequence>
              <element name="start_time"/>
              <element name="end_time"/>
              <element name="duration"/>
      </sequence>
  </model>
</elementtype>

<elementtype name="start_time">
     <instanceof name="time_val"/>
</elementtype>

<elementtype name="end_time">
     <instanceof name="time_val"/>
</elementtype>

<elementtype name="duration">
     <instanceof name="time_val"/>
</elementtype>

<elementtype name="time_val">
  <model>
    <choice occurs=1>
        <element name="frame_num"/>
        <element name="SMPTE"/>
        <element name="abs_time"/>
    </choice>
  </model>
</elementtype>
```

```
<elementtype name="frame_num">
  <model>
        <string datatype="frame"/>
  </model>
</elementtype>

<datatype name="frame">
    <scalar datatype="int" min="1" max="25"/>
</datatype>

<elementtype name="smpte">
  <model>
    <string>
        <mask>##:##:##;##</mask>
    </string>
  </model>
</elementtype>

<elementtype name="abs_time">
  <model>
        <string datatype="time"/>
  </model>
</elementtype>

<elementtype name="key_frame">
  <model>
        <string datatype="URI"/>
  </model>
</elementtype>

<elementtype name="camera_dist">
    <model>
        <string datatype="camera_distances" />
    </model>
</elementtype>

<datatype name="camera_distances">
    <enumeration datatype="nmtoken">
        <option>close-up</option>
        <option>medium-shot</option>
        <option>long-shot</option>
    </enumeration >
</datatype>

</schema>
```

7.2 Advantages of SOX for Video Metadata

- XML query languages, when available, will work on SOX documents.
- XML parsers will work on SOX documents. In order to perform complete validation of SOX-specific constraints, extra parsing code will be required.
- Provides much better data typing capabilities than the other schemas - scalar, enumerated and formatted data types.
- SOX provides the best cardinality with "occurs n,m".
- Inheritance provides the possibility for reuse of code, elements and datatype definitions. Elements can inherit their definitions from existing elements (using *instanceof*) and also extend with new attributes (using *extends*).

7.3 Disadvantages of SOX for Video Metadata

- SOX was designed for validating business documents in e-commerce applications. Consequently it is more suitable for validating static forms than complex multimedia structures. It is element-focussed rather than entity-focussed.
- It only provides the implicit "contains" relationship. Structural constraints would require multiple "contains" elements.
- Inheritance is possible but users can only extend elements with new attributes. Ideally one should be able to extend elements with both new attributes and new elements.
- Not an RDF vocabulary.

---

8. Conclusions: the Ultimate Schema

None of the above schemas is ideal for describing complex multimedia documents. They all satisfy some of the requirements but fall down in other areas. None of them is designed for describing complex hierachical structures in which there are spatial, temporal, structural and conceptual relationships between the components and where these relationships map to constraints on the relative attribute values of the related components. For example, spatial relationships such as *neighbours*, *in-front-of*, *behind*, *overlapping* and *surrounds* correspond to certain constraints on the values of the shape, location or volume attributes of the related objects. Similarly temporal relationships such as *contains*, *sequential*, *parallel* and *overlapping* should be mapped to constraints on the start

time, end time and duration of the components. None of the schemas support these capabilities.

RDF Schema claims to differ from the other schemas, in that it is not a "syntactic" schema but a "semantic schema". However the sorts of machine-understandable meanings provided in the current version of RDF Schema is very limited. So the advantage of "semantic validation" is negligible. RDF Schema is good at containing and combining descriptors from different name spaces/communities but it has virtually no data typing. Data types must be defined in a separate name space. This has yet to be done but the work is intended to be done within the W3C XML Schema Working Group [10] which has recently been set up. RDF Schemas also don't easily support multi-layered hierarchical structures because of the inability to specify generic relationship types using properties and to apply these across multiple domain/range pairs. So although RDF is better than the other schemas because of its ability to specify relationships other than the implicit *child* or *contains* relationship (which is the only one that the other schemas offer), this facility is limited to a specific range and domain due to the lack of "class-specific constraints".

XML DTDs offer simplicity and fast, cost-effective development due to the ready availability of parsers, tools and applications. However, as a data modelling language, they have limited semantics, which XML-Data, DCD and SOX schemas try to expand by adding things such as strong data typing and lexical constraints.

DCD is an improvement on XML DTDs because it provides better data typing and also provides additional semantics via its RDF conformity. However it doesn't currently support inheritance - although this is a future goal.

SOX has the best data typing. It is also XML compliant so that XML parsers and XML-QL (when it becomes available) will work on it. It supports inheritance but with attribute extension only, not element extension. It is not RDF-conformant. SOX also provides the best cardinality - enabling the minimum (other than 0 or 1) and maximum number of children of an element to be specified e.g. maximum of 10 shots per scene. DCD and XML DTDs can only specify zero or more or one or more children. RDF-Schema doesn't support cardinality.

An additional desirable schema feature would be the ability to define *equivalence* relationships between attributes and define constraints based on these relationships. For example, suppose there are two attributes, *ColourText* and *ColourHistogram*. Then in an ideal schema, users would be able to define an enumerated data type for ColourText (red, yellow, green, blue etc) and for each of these possible values, a correponding permissable range of ColourHistograms would be defined. An even more complex example is the mapping of a textual description attribute to some combination of shape, colour and texture attribute value ranges. Such a schema could then be used to both validate the integrity of the input data but also automatically generate metadata where it is not provided. This ability to map from high-level features (such as text) to low level features (colour, shape, texture) is one of the requirements of the MPEG-7 DDL. It would also greatly improve the searchability of complex multimedia archives. None of the schemas examined provide such sophisticated capabilities.

Other relevant schemas not covered in this paper include: the XSchema specification [18] and XML-Data [16]. XSchema is very similar to SOX. Formerly known as XSD, XSchema began as a proposal for the representation of XML DTDs as XML documents. The advantages of using XML document syntax to describe XML document structures include the ability to browse and edit XSchemas using XML-aware tools. This can't be done on DTDs which are not pure XML documents. Although XML-Data has some very useful features, it appears to have been superseded by DCD.

8.1 The Ideal Schema for Multimedia

Based on the above analysis and comparisons, the best solution for video metadata representation is one which provides the object-oriented, semantical concepts of RDF but expresses them within an easily-understood, human-readable XML schema. We have proposed such a XML schema for the MPEG-7 DDL [19] which provides the following features:

- The semantics and object-oriented concepts of inheritance provided by RDF through *classes*, *sub-classes*, *properties* and *sub-properties*;
- The extensible data typing capabilities of SOX;
- The addition of a *relation* entity which allows spatial, temporal, structural and conceptual relations to be defined between classes and constraints on the domain, range and property values to be specified;
- The XML namespace facility;
- Cardinality;
- Temporal and spatial controls and specifications;
- Linking mechanisms which enable links between descriptions and between content and descriptions;

The two major problems associated with this proposal are that it constitutes yet another schema or language and there are likely to be quite complex extensions necessary to the basic XML parser in order to perform complete validation of all of the constraints.

The W3C XML Schema Working Group [10] is looking at a XML-based schema language which provides support for data typing and structural constraints, currently lacking in XML DTDs. Their charter includes delivering a recommendation on the best combination of DCD, XML-Data, SOX and RDF for validating document syntax. Based on the XML Schema Requirements document [20], there is a very real possibility that the schema which they develop will satisfy the majority of the MPEG-7 DDL requirements.

Acknowledgements

The authors wish to acknowledge that this work was carried out within the Cooperative Research Centre for Research Data Networks established under the Australian Government's Cooperative Research Centre (CRC) Program and acknowledge the support of CITEC and the Distributed Systems Technology CRC under which the work described in this paper is administered.

References

[1] MPEG-7, the "Multimedia Content Description Interface"
[2] Hunter J., Iannella R., "The Application of Metadata Standards to Video Indexing", Second European Conference on Research and Advanced Technology for Digital Libraries, Crete, Greece, September, 1998.
[3] Gonno Y., Nishio F., Haraoka K., Yamagishi Y., "Metadata Structuring of Audiovisual Data Streams on MPEG-2 System", Metastructures '98, Montreal, Canada, August, 1998.
[4] Nishio F., Gonno Y., Haraoka K., Yamagishi Y., "Transporting RDF Metadata Associated with Structured Contents", Metastructures '98, Montreal, Canada, August, 1998.
[5] Bhat D., "On Representing Video Structure Using RDF", Doc ISO/IEC JTC1/SC29/WG11 MPEG98/M4132, MPEG Atlantic City Meeting, October 1998.
[6] Dublin Core Home Page
[7] MPEG-7 Requirements Document V.7", Doc ISO/IEC JTC1/SC29/WG11 MPEG98/N2461, MPEG Atlantic City Meeting, October 1998.
[8] "Resource Description Framework (RDF) Model and Syntax Specification", REC-rdf-syntax-19990222, W3C Recommendation, 22 February 1999.
[9] "Resource Description Framework (RDF) Schema Specification", WD-rdf-schema-19990218, W3C Working Draft, 18 February 1999.
[10] XML Schema Working Group
[11] Saarela J., SiRPAC - Simple RDF Parser and Compiler, 25 February 1999.
[12] Extensible Markup Language (XML) 1.0, REC-xml-19980210, W3C Recommendation 10 February 1998.
[13] Deutsch A., Fernandez M., Florescu D., Levy A., Suciu D., XML-QL: A Query Language for XML, Submission to W3C, 19 August 1998.
[14] Bray T., "Adding Strong Data Typing to SGML and XML", May 1997.
[15] Document Content Description for XML, Submission to W3C, 31 July 1998.
[16] XML-Data, W3C Note, 5 January 1998.
[17] Schema for Object-Oriented XML (SOX), NOTE-SOX-19980930, Submission to W3C, 15 September 1998.
[18] XSchema Specification, Version 1.0, November 1998
[19] Hunter J., "A Proposal for an MPEG-7 Description Definition Language", P547, MPEG-7 Test and Evaluation AHG Meeting, Lancaster, February 1999.
[20] "XML Schema Requirements", NOTE-xml-schema-req-19990215, W3C Note, 15 February 1999.

Vitae

Jane Hunter is a Senior Research Scientist within the Resource Discovery Unit at DSTC, investigating international metadata standards and schemas for multimedia resources. She has extensive experience in multimedia indexing, through the development of applications using IBM's Digital Library and the DSTC's SuperNOVA project. She is currently involved in the development of the MPEG-7 Definition Description Language and is also an active participant within the Dublin Core and RDF standards communities. She received a PhD in Computer Animation from the University of Cambridge in 1994.

Liz Armstrong is the Director of the Technology Transfer and Training Unit at the Cooperative Research Centre for Distributed Systems Technology (DSTC Pty Ltd) in Brisbane, Australia. The Unit's activities are centred on the process of transferring technology from the DSTC to the Centre's participant organisations through education, training, special events and secondment programmes. Liz holds a Bachelor of Commerce from Griffith University, with majors in public policy, marketing and video production and she is currently studying for a Masters of Commerce (Information Systems) at the University of Queensland.
