INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG99/M4518
March 1999

Title: **An "Improved" Proposal for an MPEG-7 Description Definition Language (DDL)**
Submitted to: **47th MPEG Meeting, Seoul, Korea**
Authors: **Jane Hunter, DSTC**

1. Introduction

At the recent MPEG-7 AHG Test and Evaluation Meeting in Lancaster in February, the MPEG-7 Description Definition Language (DDL) evaluation team compared and evaluated ten DDL proposals. A summary report was produced which includes a list of recommendations and outlines the technologies which should be included, considered, or further investigated for possible inclusion within the DDL for the MPEG-7 Experimentation Model (XM) [1].

The DDL Evaluation team came to the conclusion that none of the proposals sufficiently met the DDL requirements. The proposals themselves exposed the critical issues, in particular object oriented expression of semantics versus validation of structure. Most of the proposals focussed on one of these approaches and tried to extend it towards the other but it became clear that no existing language implementation can adequately reconcile these disparate requirements.

However, there was a consensus that XML be used as the syntax for the MPEG-7 DDL. There was also a consensus that there is a need for object oriented expression of semantics and validation of structural, relational and data typing constraints. As a consequence of this, the development of the DDL will be a collaborative effort to design and implement an XML-based language bridging these two approaches. P547 [2] was selected as the candidate upon which the language design will be based. In addition, the strategy is to continue tracking and influencing related efforts in the W3C community, in particular the XML Schema Working Group [3].

This submission is a modified version of the MPEG-7 DDL proposal P547, presented at the Lancaster MPEG-7 Test and Evaluation meeting [2]. It represents a first attempt to incorporate the changes which were suggested by the DDL Evaluation Team in Lancaster and also attempts to improve certain aspects which were determined to be incomplete. In particular, the following changes and suggestions have been addressed:

- Allowing property types to be classes;
- Extending data types to include certain suggested spatial types;
- Clarification of how the validation of constraints will be implemented within the parser.

It is hoped that the result is a schema which comes closer to meeting the requirements of the DDL as described in the MPEG-7 Requirements Document [4] and which can be used as a starting point for future collaborative development. This paper is informed by all of the DDL proposals submitted to the DDL Evaluation team [5-13] as well as suggestions and recommendations from the DDL evaluation team.

2. DDL Requirements

According to the MPEG-7 Requirements Document [4], the DDL requirements are:

1. **Compositional capabilities:** The DDL shall supply the ability to compose a DS from multiple DSs.
2. **Platform independence:** The DDL shall be platform and application independent. This is required to make the representation of content as reusable as possible even on grounds of changing technology.
3. **Grammar:** The DDL shall follow a grammar that is unambiguous, and allow easy parsing (interpretation) by computers.
4. **Primitive data types:** provide a set of primitive data types, e.g. text, integer, real, date, time/time index, version, etc.
5. **Composite datatypes:** The DDL must be able to succinctly describe composite datatypes that may arise from the processing of digital signals (e.g., histograms, graphs, and rgb-values).
6. **Multiple media types:** The DDL must provide a mechanism to relate Ds to data of multiple media types of inherent structure, particularly audio, video, audio-visual presentations, the interface to textual description, and any combinations of these.
7. **Partial instantiation:** The DDL shall provide the capability to allow a DS to be partially instantiated by descriptors.
8. **Mandatory instantiation:** The DDL shall provide the capability to allow the mandatory instantiation of descriptors in a DS.
9. **Unique identification:** The DDL shall provide mechanisms to uniquely identify DSs and Ds so that they can be referred to unambiguously.
10. **Distinct name spaces:** The DDL shall provide support for distinct name-spaces. Note: Different domains use the same descriptor for different features or different purposes.
11. **Transformational capabilities:** The DDL shall allow the reuse, extension and inheritance of existing Ds and DSs.
12. **Relationships within a DS and between DSs:** The DDL provides the capability to express the following relationships between DSs and among elements of a DS and express the semantics of these relations
    a) Spatial relations
    b) Temporal relations
    c) Structural relations
    d) Conceptual relations
13. **Relationship between description and data:** The DDL shall supply a rich model for links and/or references between one or several descriptions and the described data.
14. **Intellectual Property Management:** The DDL shall provide a mechanism for the expression of Intellectual Property Management and Protection (IPMP) for description schemes and descriptors.

In addition, we consider the following criteria to be important :

- Human readability (desirable but not mandatory).
- Cardinality - the ability to specify that a property, relation or attribute can have zero, one or multiple values. For example, each scene must contain between 1 and 20 shots.
- Ability to specify constraints on attribute values of related elements e.g. the start and end times of shots within a scene must lie within the start and end times of the enclosing scene. This is desirable rather than mandatory.
- Availability of supporting technologies such as parsers, databases and query languages.
- Ability to link between descriptions, description parts and between content and descriptions - Xlinks, XPointers and temporal and spatial locators.
- Spatial data types and coordinate space support.
- A mechanism for linking to ontologies.
- Real-time support - this existing requirement needs clarification. Some people thought it meant real-time generation of MPEG-7 descriptions and others assumed it meant real-time streaming of MPEG-7 descriptions.
- Mapping of Ds and DSs to MPEG-4 BIFs. It is debatable whether this is a DDL issue.
- A mechanism or interface for linking from the DDL to remote procedural code and vv (SAX or DOM).
- Validation of constraints i.e. a parser capable of validating the following: values of properties; structures, related classes and values of properties of related classes.

3. The Chosen Description Scheme

Figure 1 shows the logical structure, the structural components and their associated Dublin Core (DC) [14] attributes for a video description scheme which we use to demonstrate our schema's capabilities.

Our description scheme extends certain Dublin Core elements to cope with video content metadata requirements. This 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 cross-media searching for complete multimedia documents. Extensions or qualifiers on specific DC elements can be applied at the lower levels (scenes, shots, frames) to provide fine-grained discipline-specific searching (e.g. Description.Camera.Angle). The DC elements which are extended to provide complete video descriptions are: Type, Description, Format, Relation and Coverage. Details of the extensions are described in [15].

To represent the video structure and Dublin Core descriptors in Figure 1, a suitable schema must be able to support the hierarchical structure definition in which complete multimedia documents sit at the top level. These contain audio and video components. The video component contains sequences, which contain scenes, which contain shots, which contain frames, which contain objects or actors. In addition, each level (or class) within the hierarchy must be constrained to possess only specific properties or attributes. For simplification and because DC qualifiers are still unstable, we assume that each layer possesses both 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. These requirements are compatible with the DDL requirements listed above.

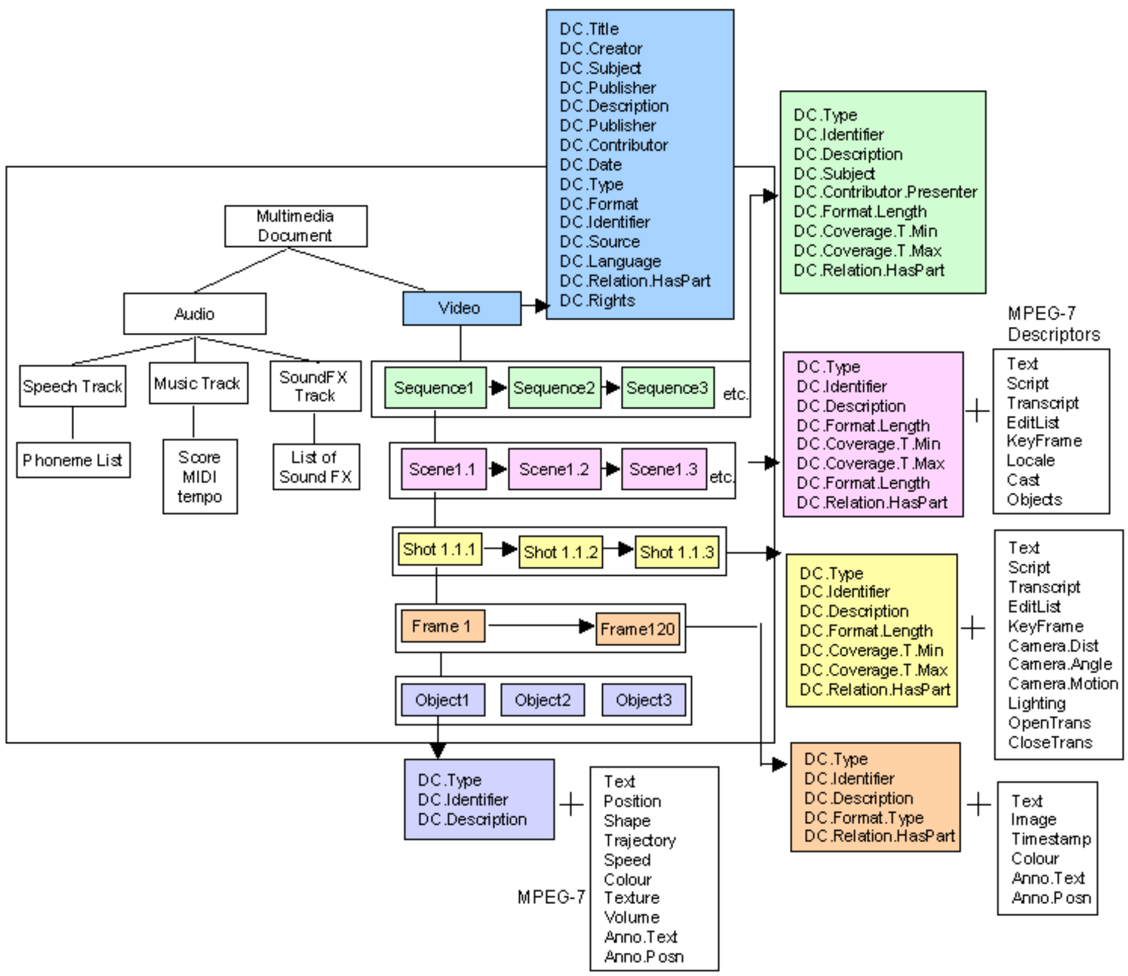

Figure 1: Multilayered Hierarchical Structure and Attributes of Video Documents

4. Proposed MPEG-7 DDL

A previous analysis of available schemas (RDF, XML DTDs, DCD, SOX) [16] found that none of the schemas was ideal for describing multimedia content. The proposal outlined in this paper takes the optimum capabilities of each of the schemas and combines and extends them where necessary, to satisfy all of the requirements of the MPEG-7 DDL. The proposed schema is based on a model consisting of classes, properties and relations between classes. It uses the classes and properties model of RDF but adds relations, timing and spatial controls and data typing capabilities. This schema provides the ability to define generic relationships and apply them with constraints on the attribute values of the related classes. It is also capable of using these definitions to validate input descriptions.

Below is a summary of the different features included within the proposed schema.

4.1 Namespace Declarations

The XML namespace facility [17] enables the inclusion of 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 accessibility and re-usability of video content on the Web. The proposed MPEG-7 schema can also be included in other schemas or descriptions using this same facility. This is a key requirement of the MPEG-7 DDL.

```
<?xml version="1.0"?>
<mpeg7
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:dc="http://purl.org/metadata/dublin_core#"
  xmlns:dcq="http://purl.org/metadata/dublin_core_qualifiers#"
  xmlns:smil ="http://www.w3c.org/TR/WD-smil#">
.
.
```

4.2 The CLASS Type Declaration and Class Hierarchies

RDF is based on a class and property data model. A collection of classes and the definition of their properties and corresponding semantics represent an RDF schema. 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. In other schemas and XML DTDs, classes and properties are not really differentiated and are all considered to be "elements".

MPEG-7 classes are equivalent to RDF classes but within each class definition, the class's properties and relations are also declared. This is more in line with XML DTDs and provides much easier readability than RDF schemas which are property-centric rather than class-centric. Using this approach, it is easier to see which properties are associated with each class.

The subClassOf property indicates the subset/superset relation between classes. It is based on the rdfs:subClassOf property. It is transitive, so that if class A is a sub-class of some broader class B, and B is a sub-class of C, then A is also implicitly a sub-class of C. Consequently, resources that are instances of class A will also be instances of C, since A is a sub-set of both B and C. Only instances of Class can have the subClassOf property and the property value is always of type Class. A class may be a sub-class of more than one class. A class can never be declared to be a sub-class of itself, nor of any of its own sub-classes. Properties are inherited from classes to subclasses. But additional properties can be added to extend subclasses. To simplify the validation, we propose that inheritance be restricted to single inheritance only i.e. each class can have a maximum of one superclass.

```
<class id="MM_Document">
   <property type="#dc_attribs"/>
</class>
<class id="Video_Document">
   <subClassOf type="#MM_Document"/>
   <property type="duration"/>
</class>
```

4.3 The Property Type Declaration

Within RDF schemas, property type definitions specify the allowable domain(s) and range. This paper proposes that a schema is easier to build, read, understand and translate to an XML DTD if the domain is defined by putting the property types in the relevant class definitions and replacing the range with either data type or class type constraints in the property definition.

Properties are defined using *propertyType*. Properties can be defined as combinations of other properties, a data type or a class type. The only allowable attributes of *propertyType* are *id*, *datatype* and *classtype*. If a datatype is specified, then the attributes *min, max, minexclusive, maxexclusive* and *default* can also be specified, if they are relevant to the datatype. Details of allowable data types are given in Section 3.7.

Properties can also be declared to be inherited from or a specialization of other existing properties using *subPropertyOf*. This is the same as the RDF subPropertyOf. If some property P1 is a subPropertyOf another more general property P2, and if a resource A has a P2 property with a value B, this implies that the resource A also has a P1 property with value B.

```
<propertyType id="timeStamp">
   <Alt>
        <property type="#SMPTE"/>
        <property type="#frame_num"/>
        <property type="#secs"/>
   </Alt>
</propertyType>

<propertyType id="startTime">
   <subPropertyOf type="#timeStamp"/>
</propertyType>

<propertyType id="endTime">
   <subPropertyOf type="#timeStamp"/>
</propertyType>

<propertyType id="SMPTE" datatype="dateTime"/>
<propertyType id="frameNum" datatype="int"/>
<propertyType id="secs" datatype="float"/>

<propertyType id="keyFrame" classtype="#Frame"/>

<propertyType id="dc_attribs">
   <property type="dc:title" occurs="zeroormore"/>
   <property type="dc:creator" occurs="zeroormore"/>
   <property type="dc:subject" occurs="zeroormore"/>
   <property type="dc:description" occurs="zeroormore"/>
   <property type="dc:publisher" occurs="zeroormore"/>
   <property type="dc:contributor" occurs="zeroormore"/>
   <property type="dc:date" occurs="zeroormore"/>
   <property type="dc:type" occurs="zeroormore"/>
   <property type="dc:format" occurs="zeroormore"/>
   <property type="dc:identifier" occurs="zeroormore"/>
   <property type="dc:source" occurs="zeroormore"/>
   <property type="dc:language" occurs="zeroormore"/>
   <property type="dc:relation" occurs="zeroormore"/>
```

The actual association of properties with classes or within other properties, is specified using the *property* element. Permissable attributes of *property* are : *type* (mandatory) and *occurs* (optional). The *occurs* attribute specifies the occurrence of properties and can have the values: *required* (one only), *optional* (zero or one), *zeroormore*, *oneormore*, *(n,m).* It is also possible for properties to be combined into groups (of order Seq, Alt, Bag or Par) within a propertyType.

4.4 The Relationship Type Declaration

One of the major advantages of RDF Schema is that its additional semantics enable relations other than "contains" to be defined. The other schemas only provide the inherent "contains" relation. However, this advantage is then weakened by the need to specify a single range constraint. Since each property can only have a single range, multilayered structures can't be described using a single generic "contains" property. This requires multiple specific "contains" properties i.e. "contains_sequences", "contains_scenes", "contains_shots", "contains_frames", each with their own specific range constraint. One alternative is to implement code outside of the schema which understands specific descriptor semantics (e.g. DC.Relation.HasParts) and can perform the validation. Another alternative which has been proposed within the RDF comments mail archive [27] is to enable class-specific constraints on properties. Class-specific constraints allow the range to be specified with the property or relation inside the class definition. We propose using this approach.

Relationships between classes can be defined using the *relationType* declaration. A *relationType* declaration must contain the name of the relation, a domain (the class to which the relation is applied) and a range (the related class). Only binary relationships are supported. In addition, it is possible to specify constraints on the properties of the related classes using the *constraint* specification. Within RDF, one cannot map relationship-type properties between classes to constraints on the property values of the classes involved. For example, 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.

Class-specific constraints are defined by inserting *relation* elements within class definitions. The domain corresponds to the container class and the range and constraint attributes can be used to override the range and constraint values in the relation definition. The range and domain for class-specific constraints must be sub-classes of the range and domain in the relationType definition.

The presence of an order attribute with the range specification implies that the range can be a group of objects of a particular class and that the group will have the specified ordering. If the occurs attribute allows groups but no ordering is specified, then any of the ordering types are permissable. If the range consists of a group, then the particular attributes are specified using an array type syntax e.g. range[1] is the first element of the group, range[n] is the last element and range[i] represents all of the elements.

The *constraint* element is used to specify constraints on the property values of the related classes. It has two attributes: *type* and *value*. The *type* can be either *boolean* or *program*. The *value* is either a logical expression between range and domain properties or a call to a remote program e.g. CORBA IDL wrapping around some external function. The program approach would be used to perform more sophisticated content checking and validation.

In addition, the relation can be defined as either uni-directional or bi-directional via attributes. If the relation is uni-directional, then an optional inverse relation name can be specified. If the relation is bi-directional, then the inverse relation is the same as the original relation.

```
<relationType id="contains" direction="uni" inverse="#contained_by">
    <domain type="#MM_Document" />
    <range type="#MM_Document" occurs="zerormore" order="Seq"/>
    <constraint type="boolean" value=
                     "((range[1].startTime>=domain.startTime)&&
                       (range[n].endTime<=domain.endTime))"/>
</relationType>

<relationType id="overlaps" direction="bi">
    <domain type="#object"/>
    <range type="#object"  occurs="zerormore" order="Bag"/>
    <constraint type= "boolean" value=
      "((domain.X.min<=(range[i].X.min +(range[i].X.max-range[i].X.min)/2)&&
        (domain.X.max>=(range[i].X.min +(range[i].X.max-range[i].X.min)/2)&&
        (domain.Y.min<=(range[i].Y.min +(range[i].Y.max-range[i].Y.min)/2)&&
        (domain.Y.max>=(range[i].Y.min +(range[i].Y.max-range[i].Y.min)/2))"/>
</relationType>

<relationType id="neighbours" direction="bi">
    <domain type="#object" />
    <range type="#object" occurs="zerormore" order="Seq"/>
    <constraint type="program" value="checkNeighbours(domain, range)"/>
</relationType>

<class id="scene">
    <subClassOf type="#MM_Document"/>
    <property type="#dc_attribs"/>
    <relation type="contains" range="#shot"/>
</class>

<class id="object">
    <subClassOf type="#MM Document"/>
```

4.5 Order and Occurs

We use the RDF Container syntax to specify sets or sequences of classes or properties. Alternatives or ordering among the elements is specified by the order attribute. RDF Container syntax provides Seq, Bag and Alt container types. We propose the addition of a Par container type for a collection of "parallel" elements. Hence the proposed values for the order attribute associated with groups are:

- Seq - an ordered list of elements or a collection of objects which occur in sequence;
- Bag - an unordered list of elements;
- Alt - a list of alternative choices from which one is possible;
- Par - a collection of objects which occur in parallel or concurrently.

Valid values for the *occurs* attribute are: *required* (occurs exactly once), *optional* (occurs zero or one times), *zeroormore*, *oneormore* and (*n,m*) (A minimum of *n* and a maximum of *m* occurences. *n* must be a positive integer or zero, *m* must be an integer greater than *n*, or "*" which indicates *m* is unbounded.)

The occurs attribute can be applied to *property*, *relation*, *group* and *range* elements.

4.6 Data Typing

If the property definition includes the *dataType* attribute, then it is a leaf node which includes the data type definition for this property.
<propertyType id="SMPTE" datatype="dateTime"/>
<propertyType id="frameNum" datatype="int" min="1" max="2000"/>
<propertyType id="secs" datatype="float"/>

In addition, it is possible to specify constraints on the content of particular properties using Min, Max, MinExclusive, MaxExclusive. Max and Min allow values upto and including the bound while MaxExclusive and MinExclusive allow values less than and greater than the bound, respectively, The semantics of upper and lower bounding are highly dependent on the element's Datatype; for some datatypes (e.g. uri), this property has no meaning.

```
<propertyType id="MonthofYear" datatype="int" min="1" max="12"/>
```

It is also possible to specify a default value using the default attribute on the property. For example:

```
<propertyType id="MonthofYear" datatype="int" min="1" max="12"  default="1"/>
```

The data type must be selected from the following:

- the intrinsic data types;
- the library of data types which have been derived from the intrinsic data types, see Appendix B;
- a user-defined data type;

XML 1.0 defines about 10 datatypes, which may only be used to constrain attribute values, and essentially one datatype, PCDATA, that can be used for element content. Here we propose a much richer set of datatypes, applicable equally to attribute and property content. The data types available are based on the SOX data typing capabilities.

The list of primitive or intrinsic datatypes are tabulated below. They are the ten XML 1.0 datatypes plus the SOX intrinsic data types.

Name	Examples	Description
id	X	XML ID
idref	X	XML IDREF
idrefs	X Y Z	XML IDREFS
entity	Foo	XML ENTITY
entities	Foo Bar	XML ENTITIES
nmtoken	Name	XML NMTOKEN
nmtokens	Name1 Name2	XML NMTOKENS
enumeration	Red Blue Green	XML ENUMERATION Legal values must be specified
notation	GIF	XML NOTATION
string	Give me liberty or give me death!	PCDATA
binary	[01]*	A sequence of bits, represented by 0 or 1.
boolean	0, 1 (1=="true")	"1" or "0"
char	char	A single character.
number	15, 3.14, -123.456E+10	A numeric value.Used when a more specific numeric representation is not required or practical. There are no contraints on the minimum or maximum values, number of digits or number of decimal places.
date	YYYY-MM-DD	A date in a subset ISO 8601 format (no time)
time	HH:MM:SS	A time in a subset ISO 8601 format, with no date and no time zone. Fractional seconds may be as precise as nanoseconds.
uri	urn:schemas-microsoft-com:Office9 http://www.ics.uci.edu/pub/ietf/uri/	Universal Resource Identifier

The enumeration data type also requires the legal values to be specified. For example:

```
<propertyType id="cameraDistance" datatype="enumeration">
   <Values>close-up medium-shot long-shot</Values>
</propertyType>
```

Jospeh Mundy [1] has suggested that the library of data types in Appendix B should be extended to provide a base set of spatial data types which includes the following: 1-D, 2-D, 3-D points, discrete point sets, curves, regions and surfaces. These have been added to Appendix B.

In addition, it has been recommended [1] that certain complex data types such as arrays, vectors and lists of given data types and sizes, should also be provided in the data type library.

4.6.1 User-defined datatypes

We propose using the same approach as the SOX W3C submission. SOX documents provide a mechanism, for defining datatypes that can be used to specify the datatype of an attribute or property content. We provide a similar mechanism through the *dataType* element. A datatype id may be referenced in the datatype attribute of the *propertyType, attributeType, enumeration, format, scalar*, and *string* elements. It is a fatal error to re-assign a datatype name or to reference a datatype that has not been defined.

User-defined datatypes may only be derived from the intrinsic datatypes. There are three types: *scalar*, *enumeration* and *format*. The valid mask values are listed in Appendix C. The MPEG-7 Schema processor must be capable of generating code to perform validation on the values of user-defined datatypes.

4.6.1.1 User-defined scalar datatypes

User-defined scalar datatypes are derived from the intrinsic number datatype. A derived datatype must specify the number of digits and decimal places, and the minimum and maximum values permitted. An optional mask describes the required format of values that conform to the datatype. The minimum and maximum permitted values may be further constrained by setting the boolean minexclusive and maxexclusive attributes to "1". A processor must be able to generate code that will validate a value against the datatype definition.

```
<dataType id="inch">
   <scalar datatype="float" digits="4" decimals="2" min="0" max="12">
     <mask>Z#.##</mask>
   </scalar>
</dataType>
```

4.6.1.2 User-defined enumeration datatypes

User-defined enumeration datatypes may be derived from any of the intrinsic datatypes. Each of the values specified in an enumerated datatype must conform to the specified type. A processor must be able to generate code that will validate the value against the datatype definition.

```
<dataType id="transition">
   <enumeration datatype="string">
      <option>cut</option>
      <option>fade</option>
      <option>wipe</option>
      <option>dissolve</option>
   </enumeration >
</dataType>
```

4.6.1.3 User-defined format datatypes

User-defined format datatypes may be derived from any of the intrinsic datatypes, but will most commonly be used to specialize string values. A required mask describes the required format of values that conform to the datatype. A processor must be able to generate code that will validate the value against the datatype definition.

```
<dataType id="part-number">
   <format datatype="string">
      <mask>AAA-###.##-aa</mask>
   </format>
</dataType>
```

4.6.2 Examples of User-defined MPEG-7 Complex Data Types

Complex data types which the DDL should be capable of supporting include : Colour histograms, 3D vectors, graphs, RGB values etc.

RGB-Values
For example, orange-red is represented by the RGB value 255;69;0 and each value must lie between 0 and 255.

```
<dataType id="rgb">
   <scalar datatype="int" min="0" max="255">
     <mask>ZZ#</mask>
   </scalar>
</dataType>

<dataType id="RGB-value">
   <format datatype="string">
      <mask>rgb;rgb;rgb</mask>
   </format>
</dataType>
```

A Colour histogram

```
<propertyType id="colourHistogram">
     <Seq> <Seq>
          <propertyType id="RGB-data" datatype="#RGB-value"/>
          <propertyType id="frequency" datatype="float"/>
    </Seq> </Seq>
</propertyType>
```

4.7 Attribute Definitions

Attributes are handled here similarly to attributes within the DCD Schema. Attributes can be specified for classes using attributeType. Properties which apply to attributeType are: id, datatype, occurs and default. *occurs* indicates whether the presence of this attribute is required. It can take one of two values: *required* or *optional.*

```
<attributeType id="duration" datatype=int occurs="optional" default="1" >

<attributeType id="employment" occurs="required" datatype="enumeration">
      <Values>Temporary Permanent Retired</Values>
</attributeType>
```

An example of the use of attribute and attributeType:

```
  <class id="MM_Document">
   <attributeType id="src" datatype="uri"/>
   <attribute type="begin" datatype="timestamp"/>
   <attribute type="end" datatype="timestamp"/>
   <attribute type="duration" datatype="timestamp"/>
```

```
  <attribute type="copyright" datatype="uri"/>
</class>
```

In this example, the properties of the attribute whose name is *src* are declared within the declaration of the MM_Document class. This would make sense if MM_Document is the only class for which the *src* attribute applies.

The second attribute, *begin*, has a declaration stored separately, referenced by its id. This declaration style is suitable when such an attribute is applicable to multiple classes; it allows maintaining the declaration in one location.

4.8 Synchronisation and Temporal Specifications

The proposal is to use similar timing controls to SMIL [26]. SMIL provides coarse-grain and fine-grain declarative temporal structuring. Coarse grain temporal information is provided by the following two structuring elements:

- <seq> ... </seq>: A set of objects that occur in sequence.
- <par> ... </par>: A collection of objects that occur in parallel.

Elements defined within a <seq> group have the semantics that a successor is guaranteed to start after the completion of a predecessor element. Elements within a <par> group have the semantics that, by default, they all start at the same time. Once started, all elements are active for the time determined by their encoding or for an explicitly defined duration. Elements within a <par> group can also be defined to end at the same time, either based on the length of the longest or shortest component or on the end time of an explicit master element. Note that if objects within a <par> group are of unequal length, they will either start or end at different times, depending on the attributes of the group.

Fine-grained synchronization control is specified in each of the continuous media object references through the following values expressed as either attributes or properties and using the comprehensive set of date and time data types provided in the library:

- *duration*=" length " attribute can be used to state the presentation time of the object;
- *begin*: specifies the explicit start of an object. It can be given as an absolute offset from the start time of the enclosing structural element by using *begin*=" time " attribute e.g. begin=="5s"; Alternatively it can be given as a relative offset to the start or end time of another sibling object using *begin*="object_id +time" attribute e.g. begin="id(x)(45s)" , begin="id(x)(end)"
- *end*: this attribute specifies the explicit end of the element

```
<seq>
  <audio_track src="audio1" />
  <audio_track begin="5s" src="audio2" />
</seq>
```

```
   audio      5s     audio
|---------|<---->|---------|
```

```
<par>
  <audio_track id="a" begin="6s" src="audio1"/>
  <img  begin="id(a)(4s)" />
</par>
```

```
        par
|-----------------|
   6s       a
<---->|-----------|
       4s
      <-->
          img
          |-------|
```

4.9 Spatial Specifications

It is possible to break an element into spatial subparts by specifying the outline of each region. In our application we define a number of methods for describing outlines. It has been recommended that spatial specifications cover 3 dimensions i.e. it should be possible to define intervals, lines, planes, polygons, bounding boxes etc. in 1D, 2D and 3D space. If a coordinate is specified as a percentage value, it is relative to the total width or height or depth of the media object display area.

Interval_1d = "left-x , right-x "
Interval_1d = "0%, 50%"
BoundingBox_2d (axis aligned) = "left-x , top-y , right-x , bottom-y"
BoundingBox_2d (axis aligned) = "0%, 0%, 50%, 50%"
BoundingPolygon_2d = "x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6…"
BoundingBox_3d (axis aligned) = "left-x , top-y , back-z, right-x , bottom-y, front-z"
BoundingPolygon_3d = "x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,x5,y5,z5,x6,y6,z6…"

4.10 Coordinate Systems

The need for coordinate systems was identified during the DDL evaluation meeting. Coordinate systems are specifications of axis, dimensions and units, used in descriptions to define n-dimensional references (e.g.: temporal, spatial, etc.). As different applications and different contexts require different notions of dimension, the DDL has to provide a way to express new coordinate systems. The mechanism for defining coordinate systems will be based on the one provided by proposal P124 [6]. The proposal is that the DDL shall provide a standard set of coordinate systems, a set of core affine transformations and the mechanism for defining transformations between coordinate systems.

Coordinate Systems:
1-D, 2-D, 3-D Euclidean
2-D Polar
3-D Cylindrical and Spherical

Transforms:
General Affine transform 1-D, 2-D, 3-D
(2x2, 3x3, 4x4 Affine Matrix specification)

5. Links

During the evaluation process, we identified the need for mechanisms which support the following types of links [1]:

- Links between parts within a single description (XLinks)
- Links between seperate descriptions (XLinks)
- Links between parts of separate descriptions (XLinks)
- Links to texts (SGML/XML) – (XPointers)
- Links to multimedia objects – SMIL-type temporal, spatial and spatio-temporal locators
- Links to fragments of multimedia objects – SMIL-type temporal, spatial and spatio-temporal locators
  For example, by using SMIL temporal anchors such as:
  <video src = "http://www.dstc/videos/98-02-20.mpg">
  <anchor id="seq3" begin="00:54:24;01" end ="00:56:32;25"/>
  </video>
  We can refer to sequence 3 by: http://www.dstc/videos/98-02-20.mpg#seq3.

Further work is required in order to clarify:
- Links to knowledge representation/ontologies
- Links from procedural code to the document and vice versa (compare the merits of SAX or DOM interfaces for multimedia documents)

6. Implementation of the Parser

The MPEG-7 parser or processor checks that the content of an MPEG-7 description conforms to the particular MPEG-7 schema being used. This includes both checking the syntax and grammar and the validation of constraints. This will require extensions to the processing code of an existing C or C++ XML Parser. (C++ is required for the Experimentation Model.) Ideally the extra validation and processing to extract the key descriptors will be implemented via a standard API. Currently there are two approaches to standard APIs for XML parsers:
- SAX (Simple API for XML) for event-based XML parsing [23].
- DOM (Document Obect Model) for tree-based XML parsing [24].

The advantages of using SAX (an event-based parser) over DOM (a tree-based parser) include:
- Tree-based APIs put a greater strain on resources (such as system memory) especially if the document is large - which is likely to be the case for multimedia documents - since the entire document is cached in memory.
- Some applications need to build their own, different data trees, and it is very inefficient to build a tree of parse nodes, only to map it onto a new tree.
- It can be very inefficient to construct and traverse an in-memory parse tree just to locate a single piece of contextual information when an event-based interface would allow you to find it in a single pass using very little memory.
- Since the DOM SDK supports SAX 1.0, it is still possible to construct and traverse a DOM in-memory parse tree using the SAX API.
- DOM is still at the proposal stage whilst SAX was released in May 1998. There are a large number of existing XML parsers and applications which support SAX.

Since the XML parser needs to be in C++ for the Experimentation Model, the choice of existing parsers is limited e.g. expat, James Clark's XML parser in C, the Language Technology Group's XML developers' toolkit LT XML or Microsoft's XML parser in C++ [25]. Unfortunately these are all non-validating parsers. Because SAX implementations are currently available only in Java and Python, it may also be necessary to write a C++ SAX driver to provide an events-based API.

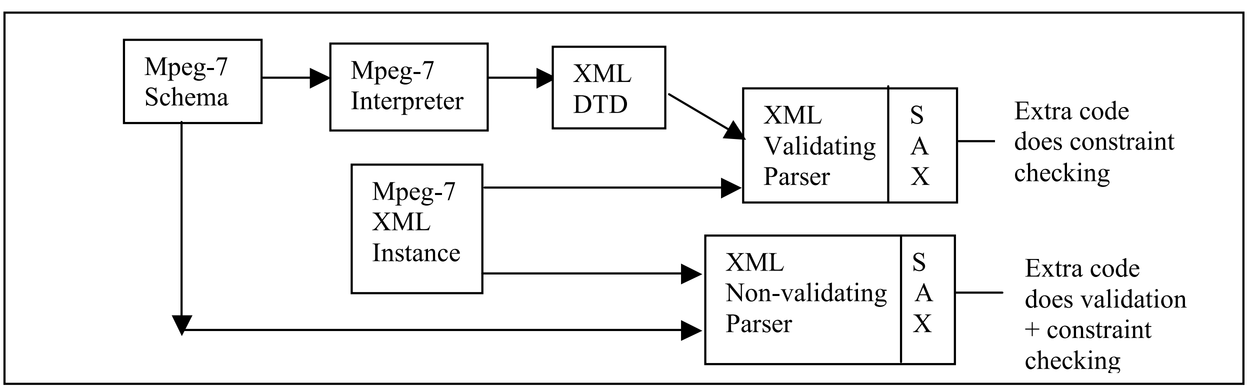

Figure 2: Processes and Software Components for MPEG-7 Parser

In addition to basic syntax and grammar checking, the parser must perform the following types of validation and return appropriate error messages when the MPEG-7 description is non-conformant:
- **Validation of Property Values**. Through the API, the processor will have to provide procedures which can convert strings (PCDATA) to each of the supported data types. It will also need to check that the given value lies within the Min and Max or MinExclusive and MaxExclusive bounds or that

the default value is substituted if no value is supplied. If a property is defined to be a particular class, then this must also be checked.

- **Validation of Cardinality**. The processor must check that all cardinality constraints are conformed to.
- **Validation of Inheritance**. Single inheritance only will be supported. Multiple inheritance can easily lead to complex inconsistencies. Inheritance will simply be provided by repetition of the properties and attributes of the superclass within the subclass and repetition of the property components in the sub-property.
- **Validation of Relations and Structures**. The processor will need to check that the properties and relations used in the MPEG-7 description all conform to the range and domain constraints specified in the MPEG-7 schema.
- **Validation of Property Constraints on Related Classes**. The processor will also need to be able to parse and evaluate the algebraic expressions used to define constraints on the property values of related classes.

7. Example of an MPEG7 Schema

Below is an MPEG-7 Schema based on the specifications above, for the Description Scheme shown in Figure 1. It is not complete but illustrates most of the features. In designing the schema, there are a number of design issues to be taken into consideration. These include:

- Deciding whether descriptors should be specified as attributes or properties. In particular the spatial and temporal controls (begin, end, duration, coords) can be either attributes or properties. The advantage of putting these values within attributes means that the overall structure is more readily apparent at first glance. We have put these as properties to ensure that they are not confused with the SMIL attributes in a SMIL multimedia presentation.
- It is also possible either to nest the descriptions of subcomponents within the descriptions of higher-level super-components or alternatively, point to the descriptions using a “meta” attribute. The meta attribute can point to either a separate file or a location within the same file.

```
<?xml version="1.0"?>
  <mpeg7
    xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
    xmlns:dc="http://purl.org/metadata/dublin_core#"
    xmlns:dcq="http://purl.org/metadata/dublin_core_qualifiers#"
    xmlns:smil ="http://www.w3c.org/TR/WD-smil#">

  <class id="MM_Document">
      <attributeType id="src" datatype="uri"/>
      <attributeType id="meta" datatype="uri"/>
      <property type="#dc_attribs"/>
      <property type="#sync_attribs"/>
      <relation type="#contains"/>
  </class>

  <relationType id="contains" direction="uni" inverse="#contained_by">
      <domain="#MM_Document"/>
      <range="#MM_Document" occurs="zerormore"/>
      <constraint type="boolean"
                  value="((range[1].startTime>=domain.startTime)&&
                          (range[n].endTime<=domain.endTime))"/>
  </relationType>

  <class id="Video">
      <subclassof="#MM_Document"/>
      <relation type="contains" range="#Sequence" occurs="zerormore"
                          order="Seq"/>
  </class>

  <class id="Audio">
```

```
  <class id="Sequence">
      <subclassof="#MM_Document"/>
      <relation type="contains" range="#Scene" occurs="zerormore" order="Seq"/>
  </class>

  <class id="Scene">
      <subclassof="#MM_Document"/>
      <relation type="contains" range="#Shot" occurs="zerormore" order="Seq"/>
      <property type="#scene_attribs"/>
  </class>

  <class id="Shot">
      <subclassof="#MM_Document"/>
      <relation type="contains" range="#Frame" occurs="zerormore" order="Seq"/>
      <property type="#shot_attribs"/>
  </class>

  <class id="Frame">
      <subclassof="#MM_Document"/>
      <relation type="contains" range="#Object" occurs="zerormore" order="Bag"
                constraint="check_contains_object(domain, range)"/>
      <property type="#frame_attribs"/>
      <property type="#visual_attribs"/>
  </class>

  <class id="Object">
       <property type="#object_attribs"/>
       <property type="#visual_attribs"/>
  </class>

<propertytype id="dc_attribs">
      <property type="dc:title" occurs="zeroormore"/>
      <property type="dc:creator" occurs="zeroormore"/>
      <property type="dc:subject" occurs="zeroormore"/>
      <property type="dc:description" occurs="zeroormore"/>
      <property type="dc:publisher" occurs="zeroormore"/>
      <property type="dc:contributor" occurs="zeroormore"/>
      <property type="dc:date" occurs="zeroormore"/>
      <property type="dc:type" occurs="zeroormore"/>
      <property type="dc:format" occurs="zeroormore"/>
      <property type="dc:identifier" occurs="zeroormore"/>
      <property type="dc:source" occurs="zeroormore"/>
      <property type="dc:language" occurs="zeroormore"/>
      <property type="dc:relation" occurs="zeroormore"/>
      <property type="dc:coverage" occurs="zeroormore"/>
      <property type="dc:rights" occurs="zeroormore"/>
  </propertytype>

<propertyType id="sync_attribs">
       <property type="begin" occurs="optional"/>
       <property type="end" occurs="optional"/>
       <property type="duration" occurs="optional"/>
</propertyType>

<propertyType id="visual_attribs">
       <property type="colourHistogram" occurs="optional"/>
       <property type="texture" occurs="optional"/>
       <property type="region" occurs="optional"/>
</propertyType>

<propertyType id="scene_attribs">
       <propertyType="transcript" occurs="optional" datatype="string"/>
       <propertyType="script" occurs="optional" datatype="string"/>
       <propertyType="edit_list" occurs="optional" datatype="string"/>
       <propertyType="locale" occurs="optional" datatype="string"/>
       <propertyType="cast" occurs="optional" datatype="string"/>
       <propertyType="object_list" occurs="optional" datatype="string"/>
</propertyType>

<propertyType id="shot_attribs">
```

```
<propertyType id="cameraDistance" datatype="enumeration">
    <Values>close-up medium-shot long-shot</Values>
</propertyType>

<propertyType id="cameraAngle" datatype="enumeration">
    <Values>low eye-level high</Values>
</propertyType>

<propertyType id="openTransition" datatype="transition">
<propertyType id="closeTransition" datatype="transition">

<propertyType id="timeStamp">
    <Alt>
         <property type="#SMPTE"/>
         <property type="#frame_num"/>
         <property type="#secs"/>
    </Alt>
</propertyType>

<propertyType id="duration">
        <subpropertyof="#timeStamp"/>
</propertyType>
<propertyType id="begin">
        <subpropertyof="#timeStamp"/>
</propertyType>
<propertyType id="end">
         <subpropertyof="#timeStamp"/>
</propertyType>

<propertyType id="SMPTE" datatype="dateTime"/>
<propertyType id="frameNum" datatype="int"/>
<propertyType id="secs" datatype="float"/>

<propertyType id="region">
    <Seq>
        <property type="#point"/>
    </Seq>
</propertyType>

<propertyType id="point" datatype="xy_pair"/>

<propertyType id="colourHistogram">
         <Seq>
              <propertyType id="colour" datatype="#rgb"/>
              <propertyType id="frequency" datatype="float"/>
        </Seq>
</propertyType>

<dataType id="xy_pair">
    <scalar datatype="string" >
      <mask>n,n</mask>
    </scalar>
</dataType>

<dataType id="rgb">
    <scalar datatype="string">
      <mask>ZZ#;ZZ#;ZZ#</mask>
    </scalar>
</dataType>

<dataType id="transition">
    <enumeration datatype="string">
        <option>cut</option>
        <option>fade</option>
        <option>wipe</option>
```

8. Instance of an MPEG-7 Video Description

Below is an example of a description of a video file, based on the MPEG-7 schema described above. In the example below, we include the Synchronized Multimedia Information Language (SMIL) namespace [26]. This allows us to refer to sequence3 by: http://www.dstc/videos/98-02-20.mpg#seq3. This is possible using SMIL temporal anchors such as:
<video src = "http://www.dstc/videos/98-02-20.mpg">
<anchor id="seq3" begin="00:54:24;01" end ="00:56:32;25"/>
</video>

This description is unnested for easy readability but the nesting of each sub-layer's description within the description of the layer above is supported. The other alternative is to use the *meta* attribute to store the descriptions in a separate file or elsewhere within the same file.

```
<?xml version="1.0" ?>
  <MPEG7
    xmlns ="http://www.w3c.org/TR/WD-rdf-syntax#"
    xmlns:dc ="http://purl.org/metadata/dublin_core#""
    xmlns:smil ="http://www.w3c.org/TR/WD-smil#"
    xmlns:dcq="http://purl.org/metadata/dublin_core_qualifiers#"
    xmnls:mpeg7="http://www.dstc.edu.au/mpeg7#">

    <MM_Document src=http://www.dstc/videos/98-02-20.mpg>
       <DC:Title>SBS World News</DC:Title>
       <DC:Subject>News, Current Affairs</DC:Subject>
       <DC:Description>Major world news events of the day.</DC:Description>
       <DC:Publisher>Special Broadcasting Service</DC:Publisher>
       <DC:Contributor.Presenter>Indira Naidoo</DC:Contributor.Presenter>
       <DC:Format DC:Scheme="IMT">video/mpg</DC:Format>
       <DC:Type>Image.Moving.TV.News</DC:Type>
       <DC:Language>en</DC:Language>
       <DC:Date>1998-05-12</DC:Date>
       <DC:Format.Length>30 mins</DC:Format.Length>
       <contains>
          <Par>
             <Seq id=="video_sequences">
                <Sequence id="seq1" src="http://www.dstc/videos/98-02-20.mpg#seq1">
                ......
                </Sequence>
                <Sequence id="seq2" src="http://www.dstc/videos/98-02-20.mpg#seq2"/>
                <Sequence id="seq3" src="http://www.dstc/videos/98-02-20.mpg#seq3"/>
                <Sequence id="seq4" src="http://www.dstc/videos/98-02-20.mpg#seq4"/>
             </Seq>
             <Seq id="audio_tracks">
                 <Audio id="music_intro" src="vivaldi.midi"/>
                 <Audio id="speech" src="naidoo.ra"/>
                 <Audio id="music_close" src="paganini.ra"/>
             </Seq>
          <Par>
       </contains>
     </MM_Document>

    <Sequence id="seq1" src="http://www.dstc/videos/98-02-20.mpg#seq1">
      <DC:Type>Image.Moving.TV.news.sequence</DC:Type>
      <DC:Description.text>"Cambodia's democracy campaigner, Sam Rainsy,
                           criticises Australia's response to his country's
                           political crisis."</DC:Description.text>
      <DC:Subject>Cambodia -- Politics, Government, History</DC:Subject>
      <DC:Contributor.Reporter>Catherine McGrath</DC:Contributor.Reporter>
      <DC:Format.Length>90 secs</DC:Format.Length>
      <DC:Coverage.t.min DC:Scheme="SMPTE">19:31:24;1</DC:Coverage.t.min>
      <DC:Coverage.t.max DC:Scheme="SMPTE">19:32:54;1</DC:Coverage.t.max>
      <contains>
        <Seq>
         <Scene id="scene1.1" src="http://www.dstc/videos/98-02-20.mpg#scene1.1"/>
```

```
<Scene id="scene1.3" src="http://www.dstc/videos/98-02-20.mpg#scene1.3"/>
   <DC:Type>Image.Moving.TV.news.sequence.scene</DC:Type>
   <DC:Description.text>"Footage of Grenade Attack" </DC:Description.text>
   <DC:Format.Length>10 secs</DC:Format.Length>
   <DC:Coverage.t.min DC:Scheme="SMPTE">19:31:44;1</DC:Coverage.t.min>
   <DC:Coverage.t.max DC:Scheme="SMPTE">19:32:07;1</DC:Coverage.t.max>
   <Transcript>"Sam Rainsy knows the violence of political life in
Cambodia. Four months ago, 16 of his supporters were killed in a grenade
attack near Phnom Penh"</Transcript>
   <Locale>"Phnom Penh"</Locale>
   <begin>"19:31:44;1"</begin>
   <end>"19:32:07;1"</end>
   <duration>"22.0"</duration>
   <contains>
       <Seq>
          <Shot id="shot1.3.1" src="http://www.dstc/videos/98-02-20.mpg#shot1.3.1"/>
          <Shot id="shot1.3.2" src="http://www.dstc/videos/98-02-20.mpg#shot1.3.2">
             ..
          </Shot>
          <Shot id="shot1.3.3" src="http://www.dstc/videos/98-02-20.mpg#shot1.3.3"/>
       </Seq>
   </contains>
</Scene>

<Shot id="#shot1.3.2" src="http://www.dstc/videos/98-02-20.mpg#shot1.3.2"/>
     <DC:Type>Image.Moving.TV.news.sequence.scene.shot</DC:Type>
     <DC:Description.text>"Woman carrying injured child."
             </DC:Description.text>
     <keyframe>" http://www.dstc/images/frame97.jpg"</keyframe>
     <cameraDistance>"close-up"</cameraDistance>
     <cameraMotion>"pan left"</cameraMotion>
     <openTransition>"cut"</openTransition>
     <closeTransition>"fade"</closeTransition>
     <begin>"19:31:48;1"</begin>
     <end>"19:32:04;1"</end>
     <duration>"15.0"</duration>
     <contains>
       <Seq>
          <Frame id="frame97" src="http://www.dstc/images/frame97.jpg"/>
          <Frame id="frame124" src="http://www.dstc/images/frame124.jpg">
             ..
          </Frame>
          <Frame id="frame165" src="http://www.dstc/images/frame165.jpg"/>
       </Seq>
   </contains>
</Shot>

<Frame id="#frame124" src="http://www.dstc/images/frame124.jpg">
     <DC:Type>Image.Moving.TV.news.sequence.scene.shot.frame</DC:Type>
     <begin>"19:31:54;1"</begin>
     <contains>
       <Bag>
          <Object id="car"/>
          <Object id="body"/>
             ..
          </Object>
          <Object id="policeman"/>
       </Bag>
   </contains>
</Frame>

<Object id="#car" src="http://www.dstc/images/frame124.jpg#car">
    <DC:Type>Image.Moving.TV.news.sequence.scene.shot.frame.object
                                                                </DC:Type>
```

9. Conclusions

We have described a schema which is capable of supporting almost all of the requirements of the MPEG-7 DDL. It is an XML schema based on the RDF concepts of classes and properties but extended with the addition of relations, context-specific constraints, data typing and cardinality. We are aware of the disadvantages of proposing a new language, but given the current limitations of XML DTDs (e.g. no data typing or inheritance), there is little alternative. We are hopeful that the XML Schema Working Group which is addressing issues such as structural schemas, inheritance, data typing and XML conformance may deliver a schema which is capable of supporting the requirements of the MPEG-7 DDL.

In the meantime, the MPEG-7 DDL WG will continue with a collaborative effort to clarify and develop the following aspects as part of the development of the MPEG-7 DDL:

- Building an MPEG-7 Schema parser/validator (in C++) for incorporation within the XM. This will require extensions to an existing XML C or C++ parser, ideally via a standard SAX or DOM API;
- Defining a base set of descriptors and description schemes and generating a library of core Ds and DSs using the DDL;
- Extending the data types to include a base set of spatial data types and defining a set of coordinate spaces and transforms;
- Clarifying and investigating the real time support requirement i.e. the generation of descriptions in real-time and/or the streaming of descriptions in real time, encoded within the content;
- Clarifying the relationship to MPEG-4 BIFs;
- Links to ontologies/knowledge bases;
- Mechanisms for linking from the descriptions to procedural code and vice versa (SAX and DOM APIs).

Finally I would like to suggest that we also investigate the XML Metadata Interchange Format (XMI) work being undertaken by the Object Management Group (OMG) [28] as there appears to be a certain amount of overlap in our objectives and the associated tools.

References

1. MPEG-7 DDL Evaluation Team Summary Report, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
2. Hunter J., DSTC , “A Proposal for an MPEG-7 DDL”, P547, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
3. XML Schema Working Group. http://www.w3.org/XML/Group/Schemas.html
4. MPEG-7 Requirements Document V.7, Doc ISO/IEC JTC1/SC29/WG11 MPEG98/N2461, MPEG Atlantic City Meeting, October 1998.
5. Kunieda T., Wakita Y., Day N., Ricoh Company Ltd, "MINDS Description Definition Language", P109, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
6. Douglass R.J., DiamondBack Systems Inc. for DARPA Consortium, "Knowledge Representation Language for MPEG-7 DDL", P124, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
7. ACTS-DICEMAN, "DICEMAN Description Definition Language", P184, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
8. Preteux F., INT, "DDL Proposal - Extendible Mark-up Language (XML)", P354, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
9. AT&T, IBM, Columbia University, "MPEG Multimedia Language (MML): A Proposal for MPEG-7 DDL", P484, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
10. McDermid E., Avid Technology, "SMPTE P18.279.981017 - Format for the Interchange of Essence and Metadata in Complex Content Packages", P486, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
11. Lennon A., Wan E., CISRA, "Dynamic Description Framework", P487, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999

12. Faudemay P., Joly P., Thienot C. and Seyrat C., Multimedia Indexing Group, LIP6, UPMC, "An Extensible DDL Framework based on RDF and Ontologies", P625, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
13. Krasinski R. and Safadi Y., Philips Research, "A Process for Describing Multimedia Content", P644, MPEG-7 AHG Test and Evaluation Meeting, Lancaster, Feb 1999
14. Dublin Core Home Page. http://purl.org/DC
15. 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. http://www.dstc.edu.au/RDU/staff/jane-hunter/EuroDL/final.html
16. A Comparison of Schemas for Dublin Core-based Video Metadata Representation, N4212, 46th MPEG Meeting, 7-11 Dec 1998, Rome
17. XML Namespaces. http://www.w3.org/TR/wd-xml-names.
18. Resource Description Framework (RDF) Schema Specification", WD-rdf-schema-19981030, W3C Working Draft, October 1998. http://www.w3.org/TR/WD-rdf-schema
19. Extensible Markup Language (XML) 1.0, REC-xml-19980210, W3C Recommendation 10 February 1998. http://www.w3.org/TR/REC-xml
20. Document Content Description for XML, Submission to W3C, 31 July 1998 http://www.w3.org/TR/NOTE-dcd
21. Schema for Object-Oriented XML (SOX), NOTE-SOX-19980930, Submission to W3C, 15 September 1998. http://www.w3.org/TR/NOTE-SOX
22. XML-Data, W3C Note, 5 January 1998. http://www.w3.org/TR/1998/NOTE-XML-Data
23. SAX 1.0: The Simple API for XML, http://www.megginson.com/SAX/index.html
24. Document Object Model, http://www.w3c.org/DOM/
25. XML Software, http://www.w3.org/XML/#software
26. Synchronized Multimedia Integration Language, WD-smil-0202, W3C Working Draft, 2 February 1998. http://www.w3c.org/TR/WD-smil
27. Brickley D., "Open Issue C23: Class-specific Constraints - Proposed Closure", http://lists.w3.org/Archives/Member/w3c-rdf-schema-wg/1998AprJun/0300.html
28. Cover R., XML Metadata Interchange (XMI), http://www.oasis-open.org/cover/xmi.html

---

Acknowledgements

Thanks to the DDL Evaluation team for their suggestions and comments. In particular I would like to thank Joseph Mundy for his enlightening philosophical discussions on RDF concepts.

Appendix A: An XML DTD for this Schema

```
<!ELEMENT class ( subclassof?|
                (property|propertyType|relation|attribute|attributeType)*)>

<!ATTLIST class id  ID  #REQUIRED>

<!ELEMENT subclassof  EMPTY>
<!ATTLIST subclassof  type CDATA #IMPLIED >

<!ENTITY %occurs_attrs
          'occurs (required|optional|oneormore|zeroormore) "required" '>
<!ENTITY %property_attrs  '(classtype?)|
                           (datatype?
                            (min|minExclusive)?
                            (max|maxExclusive)?
                            default?)'>

<!ELEMENT propertyType ( subpropertyof?|(property|Seq|Alt|Bag|Par)*)>
<!ATTLIST propertyType   id ID #REQUIRED
                         %occurs_attrs;
                         %property_attrs>

<!ELEMENT subpropertyof  EMPTY>
<!ATTLIST subpropertyof  type CDATA #IMPLIED>

<!ELEMENT property EMPTY>
<!ATTLIST property  type CDATA #IMPLIED
                    %occurs_attrs>

<!ELEMENT relationType (domain?|range?|constraint?)>
<!ATTLIST relationType id  ID #REQUIRED
                       direction (uni|bi) 'uni'
                       inverse CDATA #IMPLIED>

<!ELEMENT domain EMPTY>
<!ATTLIST domain type CDATA #IMPLIED>

<!ELEMENT range EMPTY>
<!ATTLIST range type CDATA #IMPLIED
                %occurs_attrs;
                order (Seq|Alt|Bag|Par) #IMPLIED>

<!ELEMENT constraint EMPTY>
<!ATTLIST constraint type (boolean|program) 'boolean'
                     value CDATA #IMPLIED>

<!ELEMENT relation EMPTY>
<!ATTLIST relation type  CDATA  #IMPLIED
                   range CDATA  #IMPLIED
                   constraint_type (boolean|program) 'boolean'
                   constraint CDATA #IMPLIED>

<!ELEMENT attributeType (Values?)
<!ATTLIST attributeType id ID #REQUIRED
                        %occurs_attrs;
                        %property attrs>
```

```
<!ENTITY % container "Seq|Alt|Bag|Par"

<!ELEMENT Seq ((%container;)|property|PropertyType)>
<!ATTLIST Seq id  ID #IMPLIED
              %occurs_attrs>

<!ELEMENT Alt ((%container;)|property|PropertyType)>
<!ATTLIST Alt id  ID #IMPLIED
              %occurs_attrs>

<!ELEMENT Bag ((%container;)|property|PropertyType)>
<!ATTLIST Bag id  ID #IMPLIED
              %occurs_attrs>

<!ELEMENT Par ((%container;)|property|PropertyType)>
<!ATTLIST Par id  ID #IMPLIED
              %occurs_attrs>


<!ELEMENT datatype ((enumeration|format|scalar)+ >
<!ATTLIST datatype id     ID   #REQUIRED >

<!ELEMENT enumeration (option+) >
<!ATTLIST enumeration datatype   ID           #IMPLIED
                      multiple   (true|false) "false" >

<!ELEMENT option (#PCDATA)* >
<!ATTLIST option
                value        CDATA     #IMPLIED
                label        CDATA     #IMPLIED
                selected    (selected) #IMPLIED
                disabled    (disabled) #IMPLIED >

<!ELEMENT format (mask) >
<!ATTLIST format  datatype     NMTOKEN   "string" >

<!ELEMENT scalar (mask?) >
<!ATTLIST scalar
                datatype       NMTOKEN   "number"
                digits         CDATA     #IMPLIED
                decimals       CDATA     #IMPLIED
                minvalue       CDATA     #IMPLIED
                maxvalue       CDATA     #IMPLIED
```

Appendix B: Datatype library

Derived scalar datatypes

All derived scalar datatypes have an XML parse type of string (content model of PCDATA).

byte
  Single byte. A specialization of int.
  Minimum value: -128
  Maximum value: 127
  Format: S#*
double
  A decimal value. A specialization of number.
  Minimum value: -1.17549435 * 10E308
  Maximum value: 1.17549435 * 10E308
float
  A decimal value. A specialization of number.
  Minimum value: -3.40282347 * 10E38
  Maximum value: 3.40282347 * 10E38
int
  A signed integer value. A specialization of number.
  Minimum value: -2,147,483,648
  Maximum value: 2,147,483,647
  Format: S#*
long
  A decimal value. A specialization of number.
  Minimum value: -9,223,372,036,854,775,808
  Maximum value: 9,223,372,036,854,775,807

Date and time datatypes

The following date and time datatypes are derived from ISO 8601 -- Date and Time [ISO-8601] and are informed by Date and time formats

datetime
  ISO 8601 (5.4.1.a) extended calendar date and local time format
  Format: YYYY-MM-DDThh:mm:ss
  XML parse type: string
datetime.tz
  ISO 8601 (5.4.2) extended calendar date and local time format, with time zone designator.
  Format: YYYY-MM-DDThh:mm:ssShh(:mm)?
  XML parse type: string
time.tz
  ISO 8601 (5.3.3.1) extended local time and UTC offset format.
  Format: hh:mm:ssShh(:mm)?
  XML parse type: string
time.UTC
  ISO 8601 (5.3.3) extended UTC time format.
  Format: hh:mm(:ss)?[Z]
  XML parse type: string
hour
  ISO 8601 (5.3.1.2) hour format.
  Format: hh
  Minimum: 0
  Maximum: 24
  XML parse type: string
minute
  ISO 8601 (5.3.1.4.b) minute format.

Format: -mm
Minimum: 0
Maximum: 59
XML parse type: string

second
ISO 8601 (5.3.1.4.g) second format.
Format: ss(.s)?
Minimum: 0
Maximum: 59.99
XML parse type: string

year
ISO 8601 (5.2.1.2.b) specific year format.
Format: YYYY
Minimum: 0
Maximum: unspecified
XML parse type: string

year-and-day
ISO 8601 (5.2.2.1) extended ordinal date format.
Format: YYYY-DDD
Minimum: 1
Maximum: 366
XML parse type: string

month
ISO 8601 (5.2.1.3.e) month format.
Format: --MM
Minimum: 1
Maximum: 12
XML parse type: string

day-of-a-month
ISO 8601 (5.2.1.3.d) day-of-a-month format.
Format: -MM-DD
Minimum: 1
Maximum: 31
XML parse type: string

week
ISO 8601 (5.2.3.3-Www) week format.
Format: YYYY-Www
Minimum: 1
Maximum: 53
XML parse type: string

day-of-any-week
ISO 8601 (5.2.3.3.g) day-of-any-week format.
Format: ---D
Minimum: 1 (Monday)
Maximum: 7 (Sunday)
XML parse type: string

Enumerated datatypes

countries
ISO 3166 country codes [ISO-3166]
Format: AA
XML parse type: nmtoken

currencies
ISO currency codes
Format: AAA
XML parse type: nmtoken

lang
ISO language codes [ISO-639]

Format: AA
XML parse type: nmtoken

units
ISO 31 unit identifiers [ISO-31]
Format: A+
XML parse type: nmtoken

URI datatypes

URL
Uniform Resource Locator
Format: U*
XML parse type: string

URN
Universal Resource Name
Format: [u][r][n]:U*
XML parse type: string

email
An email address
Format: [m][a][i][l][t][o]:X@X
XML parse type: string

system
XML system identifier.
Format: X*
XML parse type: string

Spatial datatypes

Points: 1-D, 2-D, 3-D

Discrete Point Sets:
- DiscretePointSet_1d
- DiscretePointSet_2d
- DiscretePointSet_3d

Curves:
- Curve_2d
- LineSegment_2d
- Ellipse_2d
- Triangle_2d
- Rectangle_2d
- Polygon_2d
- Curve_3d
- LineSegment_3d
- Ellipse_3d
- Triangle_3d
- Rectangle_3d
- Polygon_3d

Surfaces:
- Surface_3d
- Plane_3d
- Ellipsoid_3d

Appendix C: Datatype masks

A mask is a datatype format constraint. A mask consists of symbols, groups of symbols, and patterns, any of which may be modified by occurrence specifiers. Each symbol is a placeholder that stands for a character or a class of characters. Date and time masks tokens are taken from those defined in [ISO-8601]

A or a	A single alphabetic character
B or b	Any one of the boolean characters (0 or 1)
D	A single digit representing the day of the week, in the range 1-7 (Monday-Sunday) (ISO8601-5.1.2)
DD	Two digits representing a day in a month in the Gregorian calendar, in the range 01-31 (ISO8601-5.1.2)
DDD	Three digits representing a day in a year in the Gregorian calendar, in the range 001-366 (ISO8601-5.1.2)
E	The character "E", used to indicate floating point numbers
hh	Two digits representing hours in a day, in the range 00-24 (ISO8601-5.1.2)
MM	Two digits representing a month in the Gregorian calendar, in the range 01-12 (January-December) (ISO8601-5.1.2)
mm	Two digits representing minutes in an hour, in the range 00-59 (ISO8601-5.1.2)
N	Any valid XML name character
n	An integer number consisting of one or more digits.
P	The character "P", used as a "period designator" to indicate the duration of a period of time. (ISO8601-5.1.2)
p	Any one of the punctuation characters (. or : or ; or ,)
Q or q	Any one of the quote characters (" or ' or `)
S	Indicates a signed number. The characters "+" or "-" must appear in this position
ss	Two digits representing seconds in an minute, in the range 00-59 (ISO8601-5.1.2)
s	One or more digits representing a decimal fraction of a second (ISO8601-5.1.2)
T	The character "T", used as a "time designator" to indicate the start of date time of day field. (ISO8601-5.1.2)
U or u	Any character that is valid in a [URI], [URL], or [URN]
W	The character "W", used as a "week designator" to indicate the start of date week field. (ISO8601-5.1.2)
ww	Two digits representing the week number in a year, in the range 1-52 (ISO8601-5.1.2)
X or x	Any character
YYYY	Four digits of a year (ISO8601-5.1.2)
Z	The leftmost leading numeric character that can be replaced by a space character when the content of the Z position is the numeral 0
space	A single blank character
#	Any numeric character
$	A currency symbol
0	The single numeric character "0"
1	The single numeric character "1"
2	The single numeric character "2"
3	The single numeric character "3"
4	The single numeric character "4"
5	The single numeric character "5"
6	The single numeric character "6"
7	The single numeric character "7"
8	The single numeric character "8"
9	The single numeric character "9"
(...)	Represents a grouping of the symbols found between the parentheses. Within parentheses the meaning of mask symbols apply.
[...]	Represents one of the characters found between the square brackets. Within square brackets any character, except for "-" represents itself. The character "-" indicates a range of characters beginning with the character to the left of the "-" and ending with the character to the right.

*	Indicates that the preceding character, or group, may occur zero or more times.
+	Indicates that the preceding character, or group, may occur one or more times.
?	Indicates that the preceding character, or group, may occur zero or one time.
{n,m}	Indicates that the preceding character, or group, must occur at least n times and no more than m times. n must be a positive integer or zero m may be an integer greater than n, or m may be the character "*", indicating that the maximum is unbounded.
!, @, #, %, _, = /, {, }, :, ;, -,. and ,	Represent themselves