Multimedia Systems (2006) 11(4): 367–382
DOI 10.1007/s00530-006-0014-4

REGULAR PAPER

Rudinei Goularte · Maria da Graça C. Pimentel · Edson dos Santos Moreira

Context-aware support in structured documents for interactive-TV

Published online: 8 February 2006
© Springer-Verlag 2006

Abstract Interactive video technology has demanded the development of standards, techniques and tools to create, deliver and present interactive content and associated metadata. The literature reports on models representing interactive-TV (I-TV) programs – one of the main applications of the interactive video technology; the limitation of such models include: strict hierarchical relationships among media objects, low levels of granularity for metadata, programs and media objects descriptions cannot be separated, objects are not described beyond the frame level, lack of integration with context-aware computing. In this paper, we detail features of our MediaObject model for documents describing I-TV programs that aim to minimize those limitations. We discuss the model by means of an application that is part of our I-TV prototype and by an annotation tool that shows that the model can be applied to other domains.

Keywords Media descriptions · Interactive-TV · Metadata · XLink · MPEG-7 · Annotation · Reuse

1 Introduction

Interactive video technology promotes a number of applications by allowing user-interaction with objects, complex searches based on objects as well as automatic reaction in response to user-interaction. Research in this area has led to the development of standards, techniques, and tools to create, deliver and present interactive content, such as MPEG-4 [1, 2], and to provide metadata for this content, such as MPEG-7 [3, 4]. The interest shown by the communication and entertainment industries motivates the development of interactive-TV (I-TV) as one of the main applications of the interactive video technology.

The major benefit of I-TV for users is the capacity of the video delivery chain to process the digital information allowing new interaction paradigms. In particular, it is expected the construction of personalized programs, specifically designed to fit the needs of each user [5–8]. In this scenario, the structure and organization of documents containing multimedia metadata play an important role.

However, as computing becomes more pervasive and ubiquitous, users expect to be able to interact with services and applications anywhere, anytime, regardless of the device in use. Context-aware computing takes advantage of inherent context information to provide better services to the user [9, 10] – for advanced I-TV services, this implies matching the content with users' preferences or devices' capabilities. One of these services is the provision of personalized content supporting object-based interactivity. Those kinds of services need both multimedia objects to be composed by sub-objects, and documents describing the program and the components objects of the programs. Such descriptions encompass structural composition, media features (e.g. size and coding method), index information (e.g. author and creation date), links and relationships between objects and programs, and context information (e.g. the position of the objects, the location where a scene took place, or who is in the scene) [5, 7, 11, 12].

The composition of media objects using sub-objects is fully supported by the MPEG-4 standard. Moreover, MPEG-4 provides advanced standardized methods for encoding, compression, delivery and content interactivity of multimedia presentations [2]. These methods are not found in similar standards such as SMIL [13], Flash [14] or QuickTime [15]. In spite of the power for composition and segmentation of media objects provided by MPEG-4, models representing I-TV programs and related media objects reported in the literature [5, 6, 11, 16–19] present limitations that include: (a) strict hierarchical relationships among media objects, (b) low levels of metadata granularity, (c) programs and media objects descriptions cannot be separated, (d) objects are not described beyond the frame level and (e) lack of integration with context-aware computing.

We have proposed the MediaObject model to structure documents describing I-TV programs and their media

R. Goularte (✉) · M. da G. C. Pimentel · E. dos S. Moreira
Department of Computer Science, University of São Paulo at São Carlos, P.O. Box 668, 13560-970, São Carlos – SP, Brazil
E-mail: {rudinei, mgp, edson}@icmc.usp.br

objects [20]; its main features are: (a) separation of descriptions among program and media objects, (b) support to context information, (c) capability of linking descriptions between objects and programs, (d) description of in-frame objects and (e) a flexible structure for compositional media descriptions. Among other benefits, these features ease advanced content searches and reuse of objects.

In this paper, we detail how the MediaObject model can be generalized in order to also support the separation of each type of media description: structural, compositional, context and linking. Our approach segments each media into a set of MPEG-4 objects with related MPEG-7 descriptions. A main description file acts as an index to other files describing object's composition, media features, context information and linking. This allows new objects to be composed by selecting desired segments via their descriptions – the description separation feature facilitates this process. In order to support the linking of programs, objects and descriptions, we have defined an specification that uses linking information present in an instance document – this linking information is based on the XLink W3C recommendation [21]. Using XLink, we can associate an MPEG-4 media object (an in-frame object, for example) with its corresponding MPEG-7 description.

We present two applications to discuss the use of our model. The first is part of our interactive-TV prototype that illustrates in-frame interactions. The second is an annotation tool that illustrates object reuse at the same time, which shows that the model is general enough to be applied in other domains.

In this paper: Sect. 2 discusses related work towards further discussing the relevance of the overall problem tackled by our model; Sect. 3 introduces the two applications we use to discuss the MediaObject model; the definition of structured descriptions of media objects and I-TV programs is presented in Sect. 4, which details the proposed schemas and how they describe and segment media objects at the same time that represent context information; Sect. 5 explains how media objects and programs descriptions are linked and how these descriptions are associated with the corresponding MPEG-4 content; Sect. 6 presents a discussion about the present work; Sect. 7 presents final remarks and discusses future work; Appendix A gives an introduction to MPEG-4 and MPEG-7 technologies.

2 Related work

In this section, we present related works towards discussing the general problems that guided the development of the MediaObject model. A more detailed discussion and comparison with our work is given in Sect. 6.

Structured description of contents Most of the earlier approaches on describing multimedia content (including Benitez et al. [22], Dublin Core Metadata Initiative [23] and Lagoze and Hunter [24]) have been covered by the MPEG group in the form of the MPEG-7 international open ISO standard [3]. This standard, a toolbox of generic description structures with associated semantics, is meant to be instantiated by any multimedia document that requires the use of metadata. While having such schemas standardized is useful to maximize interoperability, the scopes and possible application domains are so broad that it is unrealistic to provide an exhaustive set. Consequently, MPEG-7 has been made extensible by means of its Description Definition Language (DDL) [25], allowing application developers to extend the standard description schemas towards meeting their specific requirements. Similarly, it is very unlikely that a single application will need the whole set of powerful and sometimes complex tools specified in MPEG-7. As a result, a practical way to use the standard is to make a selection of the description structures needed by the application and to validate them against specific target functional requirements.

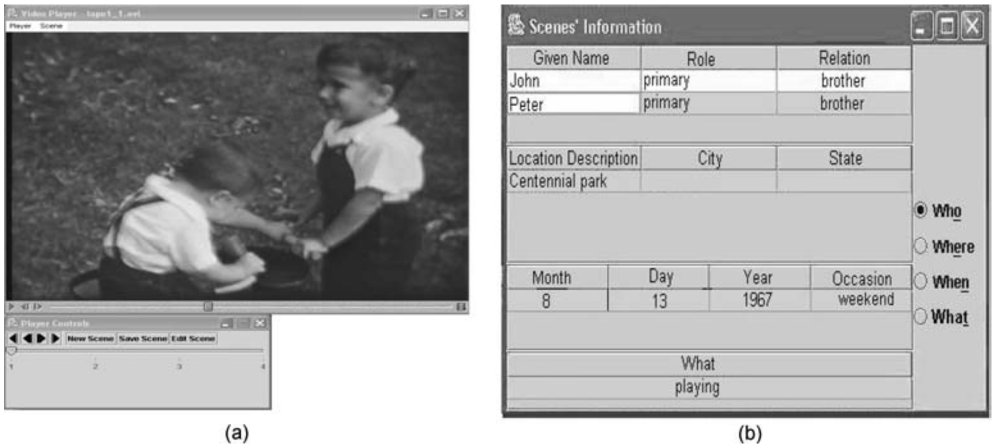
Specification of context information, reuse and addressing in-frame objects Although the problem on how to produce specific documents for the I-TV area has been addressed by many [5, 6, 11, 16, 17, 26], most of these works do not provide an adequate combination of metadata and structural representations for both programs and media objects. An important exception is the work under developement by the TV-Anytime Forum [27], which aims at standardizing interactive-TV. Their approach is to use the MPEG-7 standard to describe media objects. However, their specifications do not provide support for describing context information nor provides the means to produce independent documents for describing media objects and programs – which makes it difficult to reuse these documents. Moreover, their specifications do not make it possible to describe in-frame objects [27].

Linking programs Another problem, related to I-TV documents, is how to link programs. The TV-Anytime's solution creates an URI [27] that points to a single big object representing the program (a video, most of the times). The linking is achieved including this URI into the program's description and into program guides [27]. In this approach the program segmentation into sequences, scenes and frames is made just in a logical way, through the MPEG-7 program's media descriptions. This makes difficult to reuse program's segments and related descriptions in order to produce and to deliver personalized content. The combination of MPEG-4 and MPEG-7 standards allow the physical segmentation of programs' media with advantagens in terms of composition, interactivity and delivery. However, at the time this writting, the committees had not yet standardized a way to link an MPEG-4 object to its related MPEG-7 description.

3 The MediaObject model in use

3.1 In-frame interactions

We have built an application, part of our interactive-TV prototype, which processes a set of old home videos towards



Given Name	Role	Relation
John	primary	brother
Peter	primary	brother

Location Description	City	State
Centennial park		

Month	Day	Year	Occasion
8	13	1967	weekend

What
playing

Fig. 1 A context metadata input tool: when the user selects the "New Scene" button during a video playback **a**, a new window pops-up, allowing the user to select a context dimension to provide annotations **b**

producing what is called *programs*. The programs were segmented, coded into a number of MPEG-4 objects (352 × 288 frame size at 24 fps) and stored in a repository. An MPEG-4 object can be a whole video, a video sequence, a scene, a frame or an in-frame object.

The program and its components (MPEG-4 objects) were described according to the schemas defined in the MediaObject model. The schema instances – which are descriptions of the programs and objects – are also stored in a repository. The system uses two types of metadata: low level and high-level. The first, which includes media features such as file size, file format, file path, encoding method, frame size, frame rate, etc., can be automated. The later requires semantic knowledge about the scene – provided by the program author with the help of a context information input tool (Fig. 1).

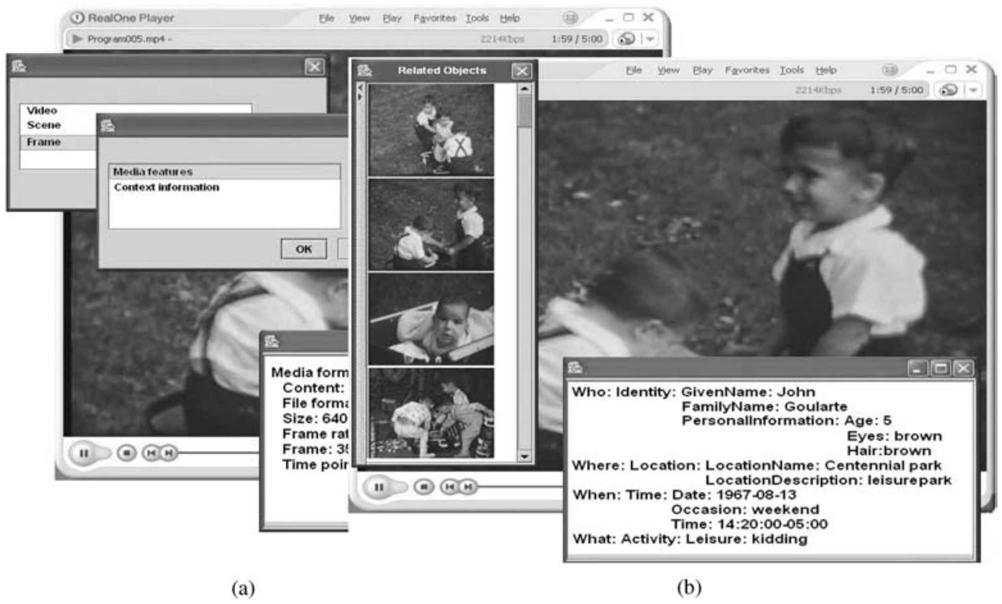
Using the MPEG-J API [2] to access the MPEG-4 objects in the scene, we have built an application that responds to mouse clicks over objects of a scene and retrieves related descriptions (features and context information relative to the object) – when a user loads a program, the descriptions (annotations) about the videos are also loaded. The implementation of this application is detailed elsewhere [20]. When a user clicks on a scene, the application allows the user choose the kind of object description to be retrieved (Fig. 2a) or, if the object is an in-frame one, the application retrieves the context information (Fig. 2b). This later case allows the user to ask the application *who* a person is just by clicking on the image. Moreover, clicking on an in-frame object makes the application to retrieve all related media objects: in the example of Fig. 2b, metadata as well as scenes and frames relative to John's brothers are retrieved along with information regarding his location.

3.2 Multimedia annotation tool

M4Note [28, 29] is a multimedia multimodal annotation tool that allows electronic ink and voice annotations on frames extracted from a video stream presented on a Tablet PCs – when a camera is used, as in Fig. 3, a video stream can be annotated at the same time that it is recorded. Moreover, the annotations can be edited, extended or played back synchronously. MediaObject is the underlying model for structured multimedia descriptions and annotations, allowing the creation of spatial, temporal and linking relationships. Our approach provides annotations as metadata for indexing, retrieval and semantic processing as well as content enrichment.

Differently from related work, M4Note allows annotations using two complementary methods: metadata association and content enrichment. Although the first approach, annotation by metadata association, has the potential to allow matching and sharing of the exact meaning of descriptions, its use can be quite arduous. However, when automatic metadata extraction and personalized tag hierarchies can be applied, it is possible to find a compromise between accuracy and effort [28, 29]. The second approach, annotation by content enrichment, may be more appealing from the user's point of view since it does not require previous knowledge of tag hierarchies and is a natural way of annotating considering annotations made by voice or ink marking. Moreover, in this case the annotation has a particular meaning for the user and as such should facilitate further understanding and retrieval.

As the user makes annotations with electronic ink marks, the system generates MediaObject instances describing the annotations as objects (for instance the *asterisk* and the *arrow* in Fig. 3). These annotation objects are included as component objects of the annotated object – the frame showing the bear in Fig. 3. In this way, the video frame is now composed by itself plus the asterisk and arrow annotation objects. Each user has his own "mark vocabulary" managed by the tool: a registration is requested at the first time the user writes a new mark so that, in the following times, the system recognizes the mark and reuses the previously registered annotation object in order to build a new object – an annotated video frame.



```
Who: Identity: GivenName: John
               FamilyName: Goularte
               PersonalInformation: Age: 5
                                    Eyes: brown
                                    Hair:brown
Where: Location: LocationName: Centennial park
                 LocationDescription: leisurepark
When: Time: Date: 1967-08-13
            Occasion: weekend
            Time: 14:20:00-05:00
What: Activity: Leisure: kidding
```

Fig. 2 **a** *Background*. Clicking on the scene causes the application to provide windows (*at the top*) allowing the user to select, first, the type of media (Video, Scene or Frame) and, next, the type of description (Media features or Context information). As a result, the application retrieves the related media features description. **b** *Foreground*. In-frame interaction and service suggesting related objects. Clicking on the taller child image, the application will retrieve the context description about that object (John's information), presenting it inside a window. Based on this description, the application retrieves all related objects where "John" matches (objects' list at left)

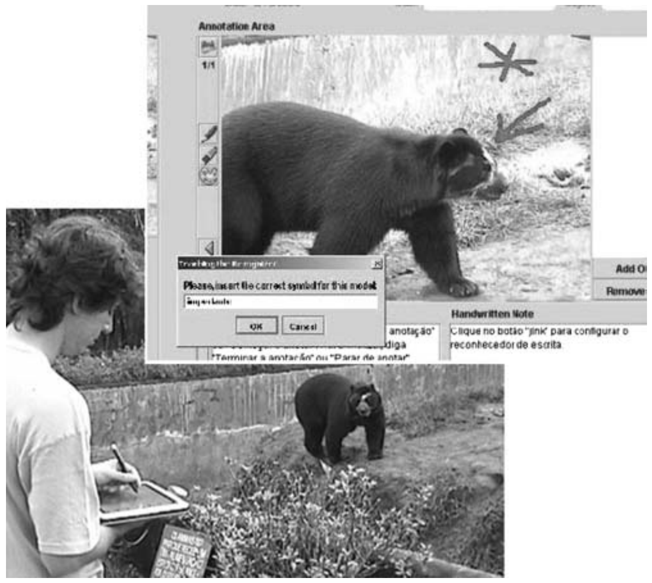

Annotation Area
Please, insert the correct symbol for this model
OK
Cancel
Handwritten Note
Clique no botão "link" para configurar o reconhecedor de escrita
Add
Remove

Fig. 3 *Background*. A user in a capture session with the M4Note running on a *Tablet PC* with an attached USB WebCam. *Foreground*. Detail of the *M4Note* tool's GUI during the definition of a new symbol: when finds an unknown symbol, the M4Note tools requires its registration

The same concepts taken from the object-based interactivity of the I-TV prototype and from object reuse achieved in the M4Note application can be applied to build other types of interactive TV programs. News programs are good examples since they are generally well structured which facilitates segmentation. In the next section we present the MediaObject model schemas, which give necessary support to building applications with context-aware features and object-based interactivity and reuse.

4 Descriptions: context and media objects

4.1 A framework for context representation

Abowd et al. [9] argue that, without good representations for context, application developers are left to develop ad-hoc and limited schemas for storing and manipulating such key information. Trying to ease the understanding and use of context, Dey [10] discusses a categorization in five dimensions – *who, where, when, what* and *why* – the first four dimensions represent, respectively, the primary types of context – *identity*, *location*, *time* and *activity*, and can be used to determine the last dimension: *why* a situation is happening.

Developers have widely used Dey's categorization of context in order to analyze application requirements. By doing so, developers can decide how to model and to represent context information for each dimension. The requirements for context information differ from one application to another, making it very difficult to cover all the possibilities of context in a self-contained document model. A similar problem was addressed by the MPEG-7 standard in the multimedia metadata representation (as discussed in Sect. 2) and, similarly to the solution adopted to MPEG-7, a solution to the representation of context information is to provide a framework with an extensible library of context elements.

In this way, developers use the framework and decide *how much* information is needed in order to represent some context entity. Moreover, developers can use the extension facilities provided to build specific context elements.

In order to guide the development of our framework, we have followed Dey's widely accepted definition for context: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves" [10]. Following this definition, developers should characterize the situation of an entity using context information. Towards allowing this characterization, our framework provides: (a) a representation for context; (b) the means to define contextual entities classifying them according to the context framework model and (c) an extensible set of pre-defined contextual elements – the context library. Our framework was built using XML Schema [30] both to ensure MPEG-7 compatibility, since the MPEG-7 DDL is XML Schema, and to facilitate information interchange.

Context hierarchies As far as the current literature is concerned, a comprehensive classification for context is still an open question – authors focus on information relative to user and infrastructure, leaving apart information relative to system, application, domain and environment. This problem occurs, in part, because of the difficulties in defining the boundaries among those entities – so that accurate taxonomies and ontologies can be modeled. In spite of these difficulties, we have defined a framework in which context models can be represented, integrated and instantiated in the hierarchical approach indicated by Fig. 4.

In our framework, a context entity type can be defined extending PrimaryContextType. PrimaryContext organizes the contextual information library used to classify context entities. A contextual entity can be classified according to many context types by making it a component of some context type – as shown in Fig. 4, "A context entity type" is a component of "A context type". As an example, in our framework UserType is a component of UsersType and represents a contextual entity – a user. Since "A context entity type" is a composite element derived by extension from PrimaryContextType, it inherits all PrimaryContext elements and attributes.

Primary Context Following Dey's definition, the situation of an entity can be characterized by (a set of) contextual information that can be indexed by the primary context.

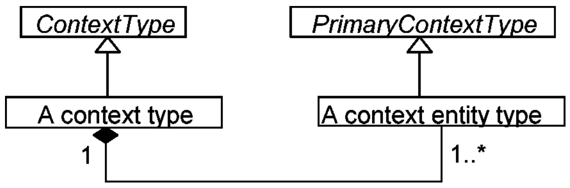


Fig. 4 Context hierarchies

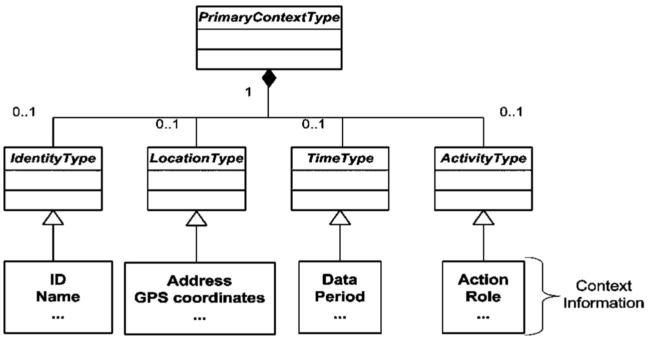


Fig. 5 Primary context information as represented in UML

In order to implement this definition, our representation of primary context is given via the abstract type PrimaryContextType that is composed by the abstract types IdentityType, LocationType, TimeType and ActivityType – which are, respectively, representations of the primary contexts *identity*, *location*, *time* and *activity* (Fig. 5).

Using these base-classes it is possible to build specialized ones in a classified fashion according to the primary context (identity, location, time or activity). We used this approach to define a context library formed by a set of these specialized classes – each library item is an XML Schema type representing context information useful to characterize an entity (the lower level in Fig. 5).

Table 1 lists some examples of the element types present in our library. Each type corresponds to one element. In the same way that complex types were built using simple types, developers can use simple or complex types in order to build their elements. For example, a type representing the users' history could be: userHistoryType = {userIDType, fullNameType, userActionType, dateType, simpleTimeType}. An exhaustive list of the types defined and detailed explanations about each element can be found at http://www.icmc.usp.br/~rudinei/Context/.

The framework aims at facilitating the developer's task: when describing a context entity, a developer chooses in the library which information best characterizes an entity. If the library does not have the right contextual information representation, developers can create their own. Since PrimaryContext type and its components are abstract, XML complex type extensions are allowed. For example, when creating a User entity through the UserType, this type must be an extension of PrimaryContextType, as in the Example 1, which implies that the UserType inherits all elements defined in PrimaryContextType (the entire context library):

```
<schema... xmlns:ctx="http://www.icmc.usp.br/~rudinei/Context/" ...
        <complexType name="UserType">
                <complexContent>
                        <extension base="ctx:PrimaryContextType"/>
                </complexContent>
        </complexType> ...
<schema>
```

Example 1

Table 1 Examples of types present in the context library

Primary context	Complex built-in types	Simple built-in types
Identity		userIDType, roleType, simpleNameType, ...
	fullNameType	givenNameType, middleNameType, familyNameType.
	personal InformationType	ageType, heightType, weightType, eyesType, hairType, birthDateType.
		locationIDType, locationNameType, locationDescriptionType, ...
Location	indoor LocationType	floorType, roomType, corridorType, gateType, exitType.
	postal AddressType	streetType, numberType, complementType, cityType, stateType, countryType, zipCodeType.
	electronic LocationType	urlType, e-mailType, phoneType, icqType.
Time		occasionType, dateType, simpleTimeType, durationType, ...
Activity		userActionType, userActivityType, ...

When describing a user, the developer chooses some pieces from the library that best match his interests. This is done using the XML xsi:type tool, the XML Schema resource used to indicate which implementation of the abstract type is being used [30]. For example, if a name is enough to describe a user, the developer decides how much information is needed to represent the name – a more structured representation composed by given name and family name (Example 2) or a single string representation (Example 3):

```
<ctx:User>
  < ctx:Identity xsi:type="ctx:fullNameType">
    < ctx:GivenName>Rudinei</ ctx:GivenName>
    < ctx:FamilyName>Goularte</ ctx:FamilyName>
  </ ctx:Identity>
</ ctx:User>
```

Example 2

```
< ctx:User>
  < ctx:Identity xsi:type="ctx:simpleNameType">
    < ctx:Name>Rudinei Goularte</ ctx:Name>
  </ ctx:Identity>
</ ctx:User>
```

Example 3

If the library does not have the right piece of information, for example in order to represent a nick name, developers extend one of the PrimaryContext components, in this case IdentityType (Example 4), so that NickNameType can be used (Example 5):

```
<complexType name="NickNameType">
 <complexContent>
  <extension base="ctx:IdentityType">
   <sequence>
    <element name="NickName" type="string"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

Example 4

```
< ctx:User>
 < ctx:Identity xsi:type="ctx:NickNameType">
  < ctx:NickName>Rudy</ ctx:NickName>
 </ ctx:Identity>
</ ctx:User>
```

Example 5

4.2 The MediaObject schema

An advanced interactive TV program needs its multimedia content to be segmented into objects and sub-objects in order to facilitate the reuse of these objects during a new program composition – but the media segmentation only it is not enough. Descriptions relative to whole program and descriptions of each component (the media objects) are necessary so that applications may match the users' interests [5, 7, 16, 26, 28, 29]. Contextual information can help this task, for example supporting the identification of *who* is in a scene or *where* the scene occurs when a user clicks on the image (see Sect. 3). However, this has being poorly explored in the interactive video domain. Moreover, reported works regarding description of multimedia objects present a lack of important features to the development of advanced interactive TV services, such as: (a) low level of metadata granularity; (b) separation of programs and media objects descriptions; (c) description of objects beyond the frame level and (d) representation of different kinds of relationships (besides the hierarchical ones only) between media objects. In this section we discuss how the MediaObject description schema overcomes the three first limitating aspects; the last one is discussed in Sect. 5.

One question could arise at this point: what kind of information need to be described? In order to match users' interests, applications need to know: the composition and the segmentation of objects and programs; the media object features and relationships between objects [5, 10, 16, 26, 31, 32]. We advocate that further information is demanded: context features. A possible solution to describe this information is the direct use of the MPEG-7 standard. However, as discussed before, applying this solution to I-TV programs will generate unnecessary complex descriptions. Besides that, MPEG-7 does not have a good set of context descriptors. Instead, our approach is to use MPEG-7 only where it is exceptionally good: in the description of low level features of audiovisual objects (details in Sect. 4.2.2).

```
1- <schema targetNamespace="urn:mpeg:mpeg7:schema:2001"
2- xmlns:Mpeg7="urn:mpeg:mpeg7:schema:2001"
3- xmlns:itv="http://www.icmc.usp.br/~rudinei/ITV/"
4- xmlns="http://www.w3.org/2001/XMLSchema"...>
 ...
5-   <complexType name="MediaObjectDescriptionType">
6-        <complexContent>
7-             <extension base="Mpeg7:CompleteDescriptionType">
8-                  <sequence>
9-                       <element name="MediaObject" type="itv:MediaObjectType"/>
10-                 </sequence>
11-            </extension>
12-       </complexContent>
13-  </complexType>
14-</schema>
                                        (a)
```

```
1- <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
2- xmlns:itv="http://www.icmc.usp.br/~rudinei/ITV/"
3- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ...>
4-   <Description xsi:type="MediaObjectDescriptionType">
5-        <MediaObject ...>
                ...
7-         </MediaObject>
8-     </Description>
9- </Mpeg7>
                                        (b)
```

Fig. 6 **a** <MediaObject> element as an MPEG-7 extension, (**b**) an instance of the MediaObject schema

This approach solves the media object features aspect and provides descriptions with a good level of metadata granularity. However, it demands a schema that describes the other aspects in a fashion that can be integrated with the MPEG-7 media features descriptions – that is, the schema must be MPEG-7 compliant. This demands the schema to be defined as an extension of one of the high level MPEG-7 tools [3, 25]. Using the MPEG-7 DDL, we defined the MediaObjectDescriptionType (Fig. 6a, line 5), which is derived by extension (Fig. 6a, line 7) from the CompleteDescriptionType MPEG-7 type [3, 25]. The MediaObjectDescriptionType is a wrap to the <MediaObject> element, which implements the MediaObject schema in the I-TV namespace (Fig. 6).

Defined in this way, the MediaObject schema becomes an MPEG-7 complete description type, ensuring MPEG-7 compliance and allowing access to the low level MPEG-7 media descriptors. In order to be instantiated, it must, first, instantiate the MPEG-7 root element (Fig. 6b). It is also necessary to indicate which type implements the complete description. In Fig. 6b, line 4, the type is MediaObjectDescriptionType.

It is now necessary to define the MediaObject elements that contain the descriptions of the mentioned aspects. We first presented the overall schema; the next two sub-sections give details regarding composition/segmentation, media and context features. Figure 7 shows a UML representation of the MediaObject description schema. The area limited by the dot-dashed lines corresponds to the main description file, where the structural composition/segmentation descriptions will be instantiated. The <MediaObject> element represents a media object, which can be just one object or can be composed by several media segments. It must be clear that the <MediaObject> element in Fig. 7 corresponds to the <MediaObject> element in Fig. 6a (line 9) and 6b (line 5).

The set of all segments that are components of a media object is represented by the <ObjectSet> element. The

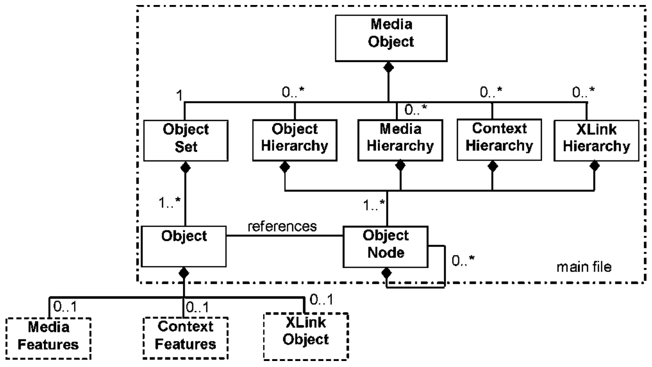
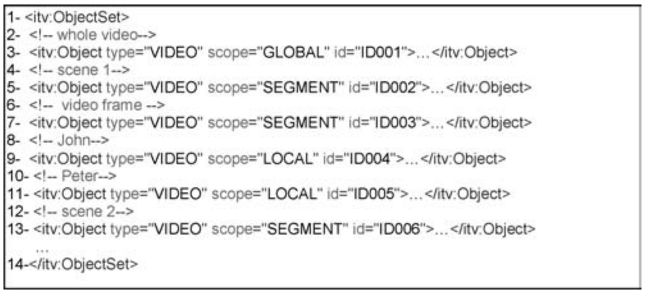


Fig. 7 UML representation of the MediaObject schema

<ObjectHierarchy> element represents the objects' composition structure, which uses <ObjectNode> elements in order to create hierarchies and make references to objects. This structural organization is based on Benitez's [22] hierarchical approach.

The advantages of the explicit use of hierarchy in a description schema include: it is a more efficient retrieval structure than graphs; a hierarchy is the most natural way to define object composition; MPEG-4 objects are built hierarchically. The same concept is applied to <MediaHierarchy>, <ContextHierarchy> and <XLinkHierarchy> elements in order to represent, in a hierarchical manner, the structure of the media, context and linking descriptions, respectively, associated with the objects.

Each object <(Object> element) may contain descriptions about its media features, context and linking, respectively <MediaFeatures>, <ContextFeatures> and <XLinkObject> elements. These elements are self-contained description schemas, and can be instantiated as separated files or inside the same MediaObject description file



```
<itv:ObjectSet>
  <!-- whole video-->
  <itv:Object type="VIDEO" scope="GLOBAL" id="ID001">...</itv:Object>
  <!-- scene 1-->
  <itv:Object type="VIDEO" scope="SEGMENT" id="ID002">...</itv:Object>
  <!-- video frame -->
  <itv:Object type="VIDEO" scope="SEGMENT" id="ID003">...</itv:Object>
  <!-- John-->
  <itv:Object type="VIDEO" scope="LOCAL" id="ID004">...</itv:Object>
<!-- Peter-->
<itv:Object type="VIDEO" scope="LOCAL" id="ID005">...</itv:Object>
<!-- scene 2-->
<itv:Object type="VIDEO" scope="SEGMENT" id="ID006">...</itv:Object>
   ...
</itv:ObjectSet>
```

Fig. 8 Segmentaton of a media object

which they refer to. The separation is possible because each element component of the <Object> was built as a MPEG-7 description unit (main file, Fig. 7). A description unit is a MPEG-7 DDL tool designed to allow pieces of descriptions to be instantiated instead of complete descriptions [3, 25].

4.2.1 Object composition

An object can be composed of (or segmented into) many other objects. For example, a video can consist of a set of scenes that can be composed of a set of frames. As illustrated in Fig. 8, each component of the segmented object is represented by an <Object> element, including the main object. All <Object> elements are encapsulated into the <ObjectSet> element and each <Object> element has the following attributes:

- Type: The object type, which can have the values VIDEO, AUDIO, IMAGE, MEDIA_FEATURES, CONTEXT_FEATURES and XLINK_FEATURES.
- Scope: The scope identifier of the object, it can have the following types:
  - GLOBAL: The object being described, which can be segmented or not.
  - SEGMENT: It is generally associated with continuous media. It is a part of the media object, a set of video scenes, for example.
  - SCENE: A scene of a video object.
  - FRAME: A video, audio or animation frame. We are using only video frames.
  - LOCAL: It is a region inside an image or frame.
  - Id: A unique identifier for objects.

It is important to observe that objects can be segmented and segments can be used in the composition of other objects – therefore an object with SEGMENT scope in one description can have GLOBAL scope in another.

The <ObjectSet> element identifies all objects and subobjects. However, it does not provide any information about how these objects are structured. That is the role played by the <ObjectHierarchy> element, for example, when describing the structural composition of the media objects illustrated in Fig. 9. This figure shows the video frame represented in Fig. 8 as the <Object> element with the id="ID003" attribute. Figure 9 illustrates how Object A is

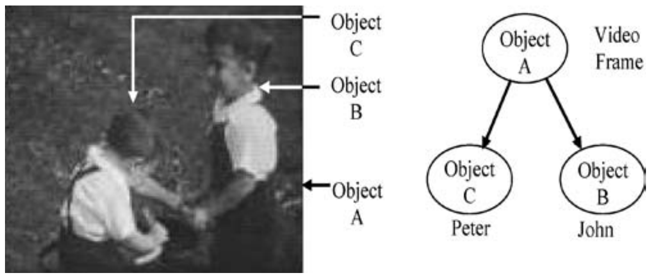
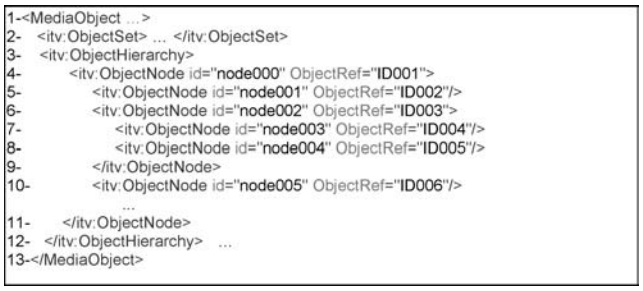


Fig. 9 A video frame composed by two in-frame objects: Peter's and John's images



```
<MediaObject ...>
  <itv:ObjectSet> ... </itv:ObjectSet>
  <itv:ObjectHierarchy>
     <itv:ObjectNode id="node000" ObjectRef="ID001">
        <itv:ObjectNode id="node001" ObjectRef="ID002"/>
        <itv:ObjectNode id="node002" ObjectRef="ID003">
           <itv:ObjectNode id="node003" ObjectRef="ID004"/>
           <itv:ObjectNode id="node004" ObjectRef="ID005"/>
        </itv:ObjectNode>
        <itv:ObjectNode id="node005" ObjectRef="ID006"/>
           ...
     </itv:ObjectNode>
  </itv:ObjectHierarchy>  ...
</MediaObject>
```

Fig. 10 Hierarchy of objects

composed of Object B, representing John, and Object C, representing Peter.

Figure 10 shows how the <ObjectHierarchy> element is used to organize the <ObjectSet> elements: each object hierarchy is a tree of <ObjectNode> elements that reference the objects of the <ObjectSet> element. Figure 10 also shows how a hierarchy allows specifying a relation of containment from the child nodes to their parent node. For instance, node002 (parent) is composed of node003 and node004 (children) which reference (ObjectRef), respectively, the objects ID003, ID004 and ID005 present in the <ObjectSet>. Moreover, these nodes represent the compositional relationship between the video frame and the "John" and "Peter" objects in Fig. 9.

Building the MediaObject schema in this way, we deal with the composition and segmentation limitations. Next we discuss how the media features and context limitating aspects are addressed by the MediaObject schema.

4.2.2 Media and context descriptions

Each object corresponds to an <Object> element (as illustrated in Fig. 7) and each of these elements may contain context and media descriptions, which correspond to the <ContextFeatures> and <MediaFeatures> MediaObject elements. We will use the example illustrated in Fig. 9 to explain how these two elements describe (a) the low level features of a media object (e.g. file type, compression and frame size) and (b) the context features about objects (like *who* is the scene).

A <MediaFeatures> element consists of a set of MPEG-7 descriptors and description schemas for audio and video (MediaFormatType); image (ImageType); and 3D objects (StillRegion3D). These descriptors and description schemas

```
<itv:ObjectSet> ...
<!-- Video frame-->
<itv:Object type="VIDEO" scope="SEGMENT" id="ID003">
  <itv:MediaFeatures>
     <Mpeg7:MediaFormat>
        <Mpeg7:Content>frame</Mpeg7:Content>
        <Mpeg7:FileFormat>MPEG-4</Mpeg7:FileFormat>
        <Mpeg7:Size>640 x 480</Mpeg7:Size>
        <Mpeg7:FrameRate variable="false">30</Mpeg7:FrameRate >
        <Mpeg7:Frame>3553</Mpeg7:Frame>
        <Mpeg7:MediaTimePoint>00:01:59</Mpeg7:MediaTimePoint>
     </Mpeg7:MediaFormat>
  </itv:MediaFeatures>
<!-- context features of the video frame-->
<itv:ContextFeatures>... </itv:ContextFeatures>
<itv:XLinkObject>... </itv:XLinkObject >
</itv:Object>
 ...
</itv:ObjectSet>
```

Fig. 11 Description of the media characteristics of an object

have been carefully selected in order to keep descriptions simple without loosing representativety and granularity when describing low level media metadata. The flexibility provided by the selected description schemas goes from the simple description of the file type (AVI, MOV, WAV, for example) to a more complete description of a video frame, such as example in Fig. 9.

The example in Fig. 9 contains three objects: Object A, representing the video frame, and Object B and Objects C, representing regions inside the video frame. A description of the physical characteristics of the video frame (Object A) is given in Fig. 11 using the <MediaFormat> MPEG-7 description schema (instantiated from MediaFormatType). Like most MPEG-7 description schemas, the MediaFormatType is a complex type. However, since most elements in that description schema are optional, there is room to make customized descriptions. The <MediaFeatures> element, employed here to describe a segment type object, may also be used to describe global objects or even local objects (such as "John" and "Peter" in Fig. 9).

When describing low level metadata of media objects, the developer's task is just to select the MPEG-7 descriptors that best match a particular object. Similarly, when describing context features of media objects, the developer must choose context descriptors, like that ones provided by the context framework discussed in Sect. 4.1. This can be done using the <ContextFeatures> element.

In the same way we have used <MediaFeatures> nested inside an <Object> element, meaning that this <MediaFeatures> element belongs to the <Object> element with ID = "ID003," we can describe the contextual characteristics of an object (in Fig. 9: Object A) by nesting a <ContextFeatures> element inside the same <Object> element (Fig. 11). However, a more interesting example of context description, which uses the <ContextHierarchy> element to describe the context characteristics of Object B in Fig. 9 (John), is given in Fig. 12.

Figure 12 shows the <ContextFeatures> element used as a wrap for the element defined in the context namespace, <Context> in the example. In order to describe the object's content features, the developer can create an <Object> element of the type CONTEXT_FEATURES, which contains the description (Fig. 12), and associate this object with the "John" object via the <ContextHierarchy> element. Figure 13 shows the node "node26" referencing John's description: this node is a child of "node25," which references the object "John," thus establishing a hierarchical relationship whereby "John's description" is contained in "John."

```
<itv:ObjectSet> ...
 <itv:Object type="CONTEXT_FEATURES" id="ID015"/>    <!-- John's description -->
   <itv:ContextFeatures>
     <ctx:Context> <ctx:User>
       <ctx:Identity xsi:type="ctx:UserInfoType">
          <ctx:GivenName>John<ctx:GivenName>
          <ctx:FamilyName>Goularte</ctx:FamilyName>
          <ctx:PersonalInformation>
             <ctx:Age>5</ctx:Age>
             <ctx:Eyes>brown</ctx:Eyes>
             <ctx:Hair>brown</ctx:Hair>
          </ctx:PersonalInformation>
       </ctx:Identity>
       <ctx:Location xsi:type="ctx:simpleLocationType">
         <ctx:LocationName>Centennial park</ctx:LocationName>
         <ctx:LocationDescription>leisure park</ctx:LocationDescription>
     </ctx:Location>
     <ctx:Time xsi:type="ctx:extendedTimeType">
        <ctx:Date>1967-08-13</ctx:Date>
        <ctx:Occasion> Weekend </ctx:Occasion>
        <ctx:Time>14:20:00-05:00</ctx:Time>
     </ctx:Time>
     <ctx:Activity xsi:type= "ctx:userActivityType">
        <ctx:Leisure>kidding</ctx:Leisure></ctx:Activity>
   </ctx:User> </ctx:Context>
 </itv:ContextFeatures>
 </itv:Object> ...
</itv:ObjectSet> ...
```

Fig. 12 Description of the context characteristics of a local object

```
<MediaObject id="MO010">  ...
    <itv:ContextHierarchy>
        <!-- John -->
        <itv:ObjectNode id="node025" ObjectRef="ID004">
            <!-- John's description -->
            <itv:ObjectNode id="node026" ObjectRef="ID015"/>
        </itv:ObjectNode>
    </itv:ContextHierarchy>
    ...
</MediaObject>
```

Fig. 13 Association between an object and its description

All information needed to describe "who John is" (Identity) and "where John is" (Location) came from the context library (observe the "ctx:" namespace identifier at the beginning of each tag) and the compositional information saying that "John's description" object belongs to (or is part of) "John" object is given by the context hierarchy described in the media object id "MO010" (Fig. 13).

In the examples given so far, the descriptions are inside the same file that defines the media object. Figure 14 illustrates another description approach using the previous context features example. The context features of the in-frame object "John" are described in a separate file. In this way, we have two files: one, instance of MediaObjectDescriptionType, defining a media object and its composition; and another, instance of ContextFeaturesDescriptionType, describing the contextual features of a specific <Object> element ("John"). The link between the context description and its related object is made via the ObjectRef attribute (Fig. 14, line 2) that refers to the object "John" ("ID004"). The file containing the definition of the object "John" must indicate which file contains John's context description – achieved via the URI attribute (Fig. 15, line 5). This approach is useful when the descriptions are voluminous, since one can organize and classify descriptions in separate files.

So far we cover the media features and context description limitating aspects, contributing to overcome the lack of structured context representation in the interactive video domain. Using object segmentation allied with good descriptions of the compositional, physical and context aspects of media objects, it is possible to facilitate the development of advanced interactive systems, like the in-frame context queries presented in Sect. 3. One last issue in these descriptions is related to the establishment of relationships between objects, beyond the hierarchical ones, and the linking of descriptions – this issue is addressed next.

5 Linking objects

5.1 Establishing relationships

An <XLinkObject> element is one of the <Object>'s components, as shown in Fig. 7, which provides support to an underlying linking structure associated with object descriptors – the aim is to allow for relationships between objects and descriptions.

As depicted in Fig. 16, an <XLinkObject> element uses the XLink standard [21] to describe the link between objects in a MediaObject description schema, to establish relationships between the objects in a MediaObject description

```
0- <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
1-          xmlns:ctx="http://www.icmc.usp.br/~rudinei/Context/"  ...>
2-   <DescriptionUnit xsi:type="ContextFeaturesDescriptionType" ObjectRef="ID004">
3-       <ctx:Context>
4-           <ctx:Users>
5-               <ctx:User>
6-                   <ctx:Identity xsi:type="ctx:UserInfoType">
7-                       <ctx:GivenName>João</ctx:GivenName>
8-                       <ctx:FamilyName>Goularte</ctx:FamilyName>
9-                       <ctx:PersonalInformation>
10-                          <ctx:Age>5</ctx:Age>
11-                          <ctx:Eyes>brown</cxt:Eyes>
12-                          <ctx:Hair>brown</ctx:Hair>
13-                      </ctx:PersonalInformation>
14-                  </ctx:Identity>
15-                  <ctx:Location xsi:type="ctx:simpleLocationType">
16-                     <ctx:LocationName>Centennial Park</ctx:LocationName>
17-                <ctx:LocationDescription>leisure park</ctx:LocationDescription >
18-                  </ctx:Location>
19-                  <ctx:Time xsi:type="ctx:extendedTimeype">
20-                      <ctx:Date>1967-08-13</ctx:Date>
21-                      <ctx:Occasion> Weekend </ctx:Occasion>
22-                      <context:Time>14:20:00-05:00</ctx:Time>
23-                  </ctx:Time>
24-                 <ctx:Activity xsi:type="ctx:userActivityType">
25-                     <ctx:Action>kidding</ctx:Action>
26-                 </ctx:Activity>
27-              </ctx:User>
28-          </ctx:Users>
29-       </ctx:Context>
30-   </DescriptionUnit>
31-</Mpeg7>
```

Fig. 14 Example of a description as a third party file

```
 ...
<MediaObject ...>
    <itv:ObjectSet>
       ...
      <itv:Object type="VIDEO" scope="LOCAL" id="ID004">
           <itv:MediaFeaturesDescription URI="./004_MediaFeatures.xml"/>
           <itv:ContextFeaturesDescription URI="./004_ContextFeatures.xml"/>
           ...
      </itv:Object> ...
    </itv:ObjectSet>  ...
</MediaObject>  ...
```

Fig. 15 Reference to the file containing the context description of the object ID004 ("John.")

```
...
<itv:Object type="VIDEO" scope="LOCAL" id="ID004">
     ...
   <XLinkObject xmlns:xlink="http://www.w3.org/2000/10/xlink-ns"
             xlink:type="extended"  xlink:title="Objects' relationships">
      <itv:Locator
         xlink:type="locator"
         xlink:href="http://www.acme.org/ITV/MO010.xml#XPointer_expression"
         xlink:role="http://www.acme.org/object"
         xlink:label="ID004"> <!-- John -->
      </itv:Locator>
      <itv:Locator
         xlink:type="locator"
         xlink:href="http://www.acme.org/ITV/MO010.xml#XPointer_expression"
         xlink:role=" http://www.acme.org/object"
         xlink:label="ID005"> <!-- Peter -->
      </itv:Locator>
      <itv:Relation
         xlink:type="arc"
         xlink:from="ID004" <!-- John -->
         xlink:to="ID005" <!-- Peter -->
         xlink:arcrole="http://www.acme.org/relations/Right_Of">
      </itv:Relation>
      ...
 </XLinkObject>
</itv:Object>
...
```

Fig. 16 Relationship between two objects

schema and to establish links between media objects and programs. Figure 16 describes the link between the objects "John" and "Peter" depicted in Fig. 9, and the relationship between these two objects – John is at Peter's right:

The <Locator> elements are locators in the context of the XLink standard and represent the "John" and "Peter" objects.

The <Locator> elements are associated with the objects through the object identifiers (IDs) as XLink labels (e.g., xlink:label="ID004").

The XLink locator type uses an XPointer expression [33] (*XPointer_expression*, in Fig. 16) to locate a specific portion of text inside a valid document (http://www.acme.org/ITV/MO010.xml, in Fig. 16), where the document is a MediaObject description schema instance.

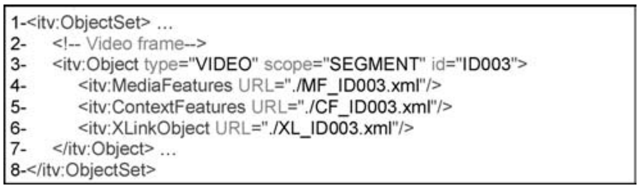
The <Relation> element describes an XLink arc from the object ID004 ("John") to the object ID005 ("Peter"). The relationship among these two objects is described by the XLink *arcrole* attribute, where object ID004 RightOf object ID005 indicates that John is at Peter's right.

In this way, a variety of relationships like temporal, spatial, directional and topological relationships can be established between objects. The link between an <Object> element and its media and context descriptions is done pointing a <Locator> element to the related <Object>'s <MediaFeatures> and <ContextFeatures> elements.

It is very important to observe that this approach allows the complete separation of link, media, context and structural descriptions.

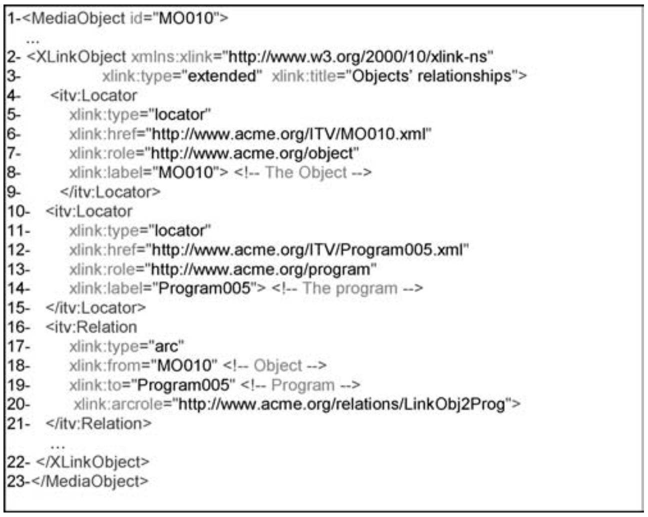
The given examples show all descriptions inside the same *MediaObject* instance XML file. However, it is possible to have each type of description in separate files. The <MediaFeatures>, <ContextFeatures> and <XLinkObject> elements, children of an <Object> element, have an optional URL attribute. This attribute points to separated XML MediaObject instances files containing only media features, context or linking information (Fig. 17). In this case, the <Locator> elements must point to these XML files and each file has references to its related <Object> (Fig. 18). This separation facilitates the reutilization of descriptions; MPEG-4 objects can be reused to compose new objects and their descriptions do not need to be edited in order to describe the components of another object.

The same approach is used to link an <Object> with its related I-TV program description, using a <Locator> element pointing to an external file. Figure 19 illustrates a case



```
<itv:ObjectSet> ...
   <!-- Video frame-->
   <itv:Object type="VIDEO" scope="SEGMENT" id="ID003">
      <itv:MediaFeatures URL="./MF_ID003.xml"/>
      <itv:ContextFeatures URL="./CF_ID003.xml"/>
      <itv:XLinkObject URL="./XL_ID003.xml"/>
   </itv:Object> ...
</itv:ObjectSet>
```

Fig. 17 Elements pointing to separated description files



```
<MediaObject id="MO010">
  ...
<XLinkObject xmlns:xlink="http://www.w3.org/2000/10/xlink-ns"
          xlink:type="extended"  xlink:title="Objects' relationships">
   <itv:Locator
      xlink:type="locator"
      xlink:href="http://www.acme.org/ITV/MO010.xml"
      xlink:role="http://www.acme.org/object"
      xlink:label="MO010"> <!-- The Object -->
    </itv:Locator>
  <itv:Locator
      xlink:type="locator"
      xlink:href="http://www.acme.org/ITV/Program005.xml"
      xlink:role="http://www.acme.org/program"
      xlink:label="Program005"> <!-- The program -->
  </itv:Locator>
  <itv:Relation
      xlink:type="arc"
      xlink:from="MO010" <!-- Object -->
      xlink:to="Program005" <!-- Program -->
       xlink:arcrole="http://www.acme.org/relations/LinkObj2Prog">
  </itv:Relation>
  ...
</XLinkObject>
</MediaObject>
```

Fig. 19 Example of linking between an object and a program

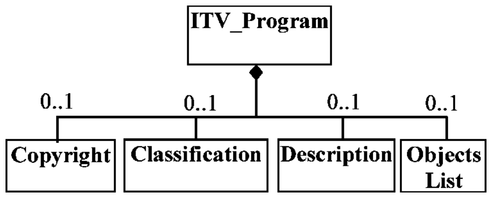


Fig. 20 UML representation of the I-TV Program schema

in which a link is established between the object identified as "MO010" ("from" attribute) and the program identified as "Program005" ("to" attribute).

The ITV_Program description schema describes I-TV programs such as News, Sports, Series, etc. Figure 20 shows a UML representation of the ITV_Program description schema. The elements Copyright, Classification and Description are imported from the TV-Anytime namespace. These elements describe the program's characteristics, such as the classification of the program (movie, news, social life, entertainment, etc.), which group this program is part of, the program's schedule, credits, reviews, genre, title, synopsis, languages, etc. A complete discussion of each of these elements and their sub-elements are outside the scope of this paper. Our aim here is to show how program descriptions are linked with media object descriptions, the linking is done via XLink, using the ObjectsList element.

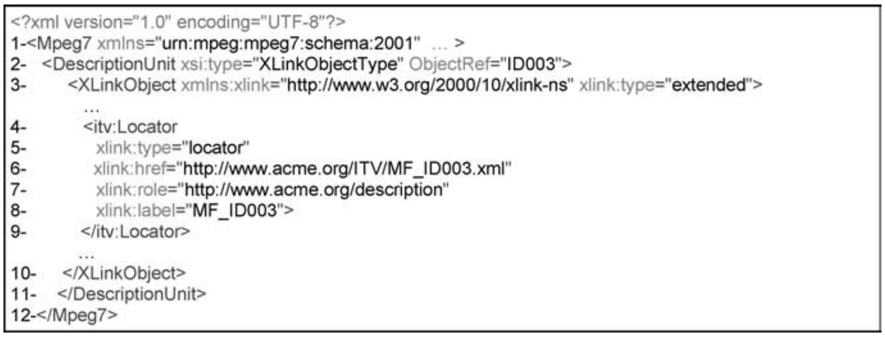
The <ObjectsList> element is the set of all links, linking the program with its media objects. In Fig. 21 the first <Locator> element represents the program and the subsequent <Locator> elements represent its objects. The <Arc> elements are XLink arcs and establish links between the programs and their component objects.

5.2 Associating MPEG-7 descriptions with related MPEG-4 media

Examining the description schemas presented in Sect. 4 and the applications examples given in Sect. 3, a question arises: how to associate MPEG-7 media object descriptions with their related MPEG-4 content?

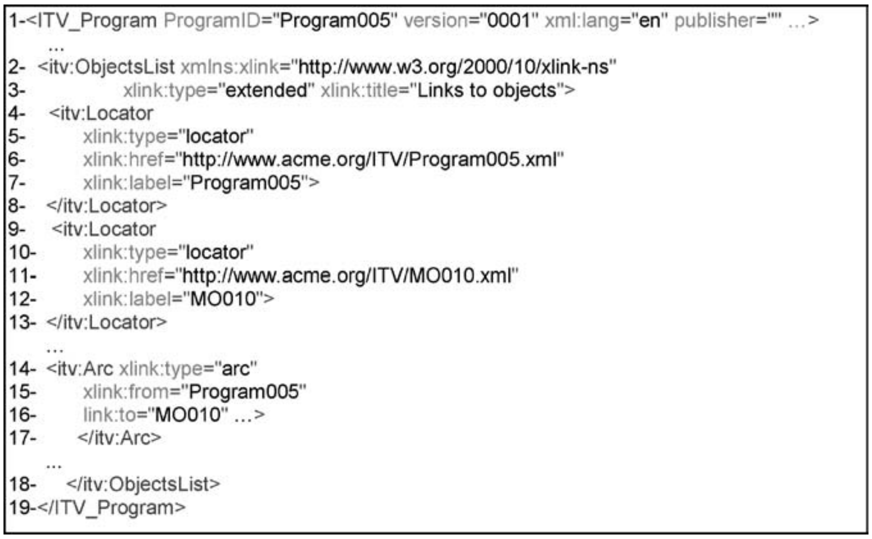
The same question is under investigation inside the MPEG-7 and MPEG-4 committees. At the time this paper was being written the committees had not published a standardized solution for this question. The ideas being discussed by the committees are about to code MPEG-7 descriptions into a binary format, and to insert these coded descriptions into a MPEG-4 presentation as a MPEG-4 elementary stream [1, 3]. This solution has its target in the efficiency of the processing and retrieving of the semantic information associated with the content, in special when working with high volumes of data. As a disadvantage, the committees' solution requires the acquisition of a MPEG-7 codec in addition to the MPEG-4 codec.

An alternative solution is to use the descriptions in their textual format (XML files), making the processing and the retrieving of semantic information via an MPEGLet [2] (see Appendix A). The technique we have developed uses



```
<?xml version="1.0" encoding="UTF-8"?>
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"  ... >
  <DescriptionUnit xsi:type="XLinkObjectType" ObjectRef="ID003">
     <XLinkObject xmlns:xlink="http://www.w3.org/2000/10/xlink-ns" xlink:type="extended">
       ...
       <itv:Locator
         xlink:type="locator"
         xlink:href="http://www.acme.org/ITV/MF_ID003.xml"
         xlink:role="http://www.acme.org/description"
         xlink:label="MF_ID003">
       </itv:Locator>
       ...
    </XLinkObject>
  </DescriptionUnit>
</Mpeg7>
```

Fig. 18 XL_ID003.xml file.



```
<ITV_Program ProgramID="Program005" version="0001" xml:lang="en" publisher="" ...>
    ...
 <itv:ObjectsList xmlns:xlink="http://www.w3.org/2000/10/xlink-ns"
             xlink:type="extended" xlink:title="Links to objects">
   <itv:Locator
        xlink:type="locator"
        xlink:href="http://www.acme.org/ITV/Program005.xml"
        xlink:label="Program005">
  </itv:Locator>
   <itv:Locator
        xlink:type="locator"
        xlink:href="http://www.acme.org/ITV/MO010.xml"
        xlink:label="MO010">
  </itv:Locator>
    ...
  <itv:Arc xlink:type="arc"
        xlink:from="Program005"
        link:to="MO010" ...>
      </itv:Arc>
    ...
    </itv:ObjectsList>
</ITV_Program>
```

Fig. 21 Link from a program to its objects

MPEG-4 elementary streams IDs as the IDs of the <Object> schema elements. In this way, the MPEGLet application obtains an object's ID during users' interactions and retrieves the associated descriptions.

The IDs association starts in the authoring phase. As discussed in Appendix A, the XMT file is where an MPEG-4 author defines a scene. To do that, the author specifies the objects present in the scene and, for each object, actions, behaviors, positions in the space and time and an ID. The XMT file is coded in a binary representation (the MPEG-4 file format .mp4) containing: the objects in the form of MPEG-4 elementary streams and a graph representation of the scene, where each node in the graph points to an elementary stream.

After the MPEG-4 authoring phase comes the description authoring phase, in which the author describes the program and the media objects using our model. Some information can be described using the tool presented in Sect. 2.1, but meanwhile the objects' IDs are assigned manually.

The final step in the ID association is done by an MPEGLet application, in three parts: a client-server socket connection, to send the descriptions to the user's machine, a monitor for the user's interactions over objects and a description parser.

As discussed in Appendix A, an MPEGLet is coded into an MPEG-4 file as an elementary stream. When the user loads the MPEG-4 video, the decoder extracts the MPEGLet elementary stream and sends it to the user's Java Virtual Machine. From this point the MPEGLet is a regular Java program. The MPEGLet we have developed establishes a socket connection with a server retrieving the description associated with the program (video) being played. The retrieving of the program's objects descriptions is done following the link information present in the program's <ObjectsList> element (see Sect. 5.1). At the user's machine, these descriptions are transformed into DOM tree representations [34] (using the Xerces API [35]), which are used in order to locate a specific element into a description and in the processing of XLink information.

Using the MPEG-J API, it is possible to manipulate the graph of the scene changing or retrieving objects' properties previously defined in the XMT file. The MPEGLet obtains the ID of a specific object every time the user clicks on this object. Based on this ID, the MPEGLet searches the DOM tree representations of the program and media object's descriptions to locate the correct <Object> element and its associated <XLinkObject> element. From this point on, an XLink parser can use the DOM trees in order to process the xlinks, locate the requested kind of description (context or media features) and retrieve information (Figs. 1 and 2). The XLink parser was built on top of the Fujitsu XLink processor [36].

6 Discussion

The problem on how to produce specific documents to Interactive TV has been addressed by many [5, 6, 11, 16, 17, 26] – however, these works do not provide adequate representations of I-TV programs and media objects in order to facilitate the development of advanced interactive services. The problem occurs, mainly, because: (a) these representations do not allow to go deep with metadata granularity in general, and with context metadata in special, limiting the precision that a media object can be described with [6, 16, 17, 26]; and (b) they do not decouple objects descriptions from the program descriptions [5, 11, 18, 27].

An important work in the I-TV area is being done by the TV-Anytime Forum [27] (see Sect. 2). TV-Anytime covers important features when describing an I-TV program, like copyright, program classification, synopsis, media content and media segmentation. Most of these items are described using the MPEG-7 standard. This model is very complete and it is a reference for I-TV development. However, their work leaves some issues that must be tackled.

First, the TV-Anytime's program concept is a monolithic object; it is not based on objects composition. A program like the "X Files" TV series can be segmented into chapters, but the chapters cannot be segmented into video sequences or scenes. Besides that, program's information like genre and synopsis are not separate from media's information like frame size and coding method. Our MediaObject model goes on other direction – programs are composed by a set of objects and objects can be segmented until the in-frame level.

Second, as TV-Anytime widely uses MPEG-7 its instances may lead to descriptions that are unnecessary complex (as discussed in Sects. 2 and 4). In contrast, our MediaObject model makes a selective use of the MEPG-7 – where it does the best job: description of low level media features. The structural information about media segmentation and composition is done in a simple way using explicit hierarchies. As an example, using MediaObject to describe the frame media features in Fig. 9 we have three descriptions levels before to reach the MPEG-7 descriptors (see Fig. 11). Using the TV-Anytime model we have six descriptions levels, and TV-Anytime does not go beyond the video level. Using the MPEG-7 pure approach, like used by Cosmas et al. [11], we have seven descriptions levels.

Last but not least, neither TV-Anytime nor MPEG-7 has structured context descriptors. MPEG-7 provides a set of seven semantic descriptors: who, what object, what action, where, when, why and how. These descriptors are, however, simple tags for textual strings, lacking a more structured organization of context information. On the other hand, our MediaObject schema uses the context framework and allows structured descriptions of context information about media objects.

The approaches adopted by the TV-Anytime and others have important impacts in the offering of advanced services such as searching for specific information and reusing objects in the composition of new ones. To illustrate the first case, consider the "content suggestion" service example given in Sect. 3. Since the MediaObject model decouples composition, media features, context and linking descriptions, it facilitates the searching process: searches can be directed to look for the information in the right place. From the example, in order to select related objects, the system retrieves the "*who*" context information about the in-frame objects and looks for matches only at other objects' context descriptions – composition and media features descriptions can be discarded since the name of person in the scene (John) is not be there. In the other approaches this kind of search is more difficult since the context information, when present, can be everywhere in the description files.

Another question is related with more specific queries, like "video objects where John is at Peter's right." Since the MediaObject model makes it possible to describe relationships (spatial, temporal, positional, etc.) between objects (<XLinkObject> element), it is simple, for instance, to restrict the query to select only objects with the target positional relationship. None of the mentioned related works discuss a similar feature.

To illustrate the second case, consider the M4Note example presented in Sect. 3. Each ink mark made by a user must be transformed into an annotation media object that must be used to enrich the content being annotated. To achieve this, the tool composes a new object – an annotated object – from the previous ones. This is possible because M4Note uses the MediaObject model to describe objects, including the annotation marks. As that model gives support to object composition and segmentation, the process of creating an annotated object is simple. In the example in Sect. 3, the object being annotated is the video frame showing a bear and the annotation marks are an asterisk and an arrow. Using our model, these annotations are included in the video frame as in-frame objects; the <Object> elements for asterisk and arrow are inserted into the <ObjectSet> element of the main video description (Example 6) and two nodes are inserted in the video frame hierarchy description (Example 7, lines 7, 12 and 13):

```
1-<itv:ObjectSet>
2-<!-- whole video-->
3- <itv:Object type="VIDEO" scope="GLOBAL"
      id="ID001">...</itv:Object>
        ...
4- <!--  video frame -->
5- <itv:Object type="VIDEO" scope="SEGMENT" id="ID010">...
6-     </itv:Object>
        ...
7-   <!-- Asterisk-->
8-   <itv:Object type="VIDEO" scope="LOCAL" id="ID016">...
          </itv:Object>
9-   <!-- Arrow-->
10- <itv:Object type="VIDEO" scope="LOCAL" id="ID017">...
          </itv:Object>
11-</itv:ObjectSet>
```

Example 6

```
1-<MediaObject ...>
2-  <itv:ObjectSet> ... </itv:ObjectSet>
3-    <itv:ObjectHierarchy>
4-      <itv:ObjectNode id="node000" ObjectRef="ID001">
5-        <itv:ObjectNode id="node001" ObjectRef="ID002"/>
6-                  ...
7-        <itv:ObjectNode id="node011" ObjectRef="ID010">
8-          <itv:ObjectNode id="node017" ObjectRef="ID016"/>
9-          <itv:ObjectNode id="node018" ObjectRef="ID017"/>
10-       </itv:ObjectNode>
11-                 ...
12-       <itv:ObjectNode id="node017" ObjectRef="ID016"/>
13-       <itv:ObjectNode id="node018" ObjectRef="ID017"/>
14-     </itv:ObjectNode>
15-   </itv:ObjectHierarchy>        ...
    </MediaObject>
```

Example 7

As a result, the video frame is now composed by itself and the asterisk and arrow annotation objects (Fig. 3). None of the mentioned related works can reuse objects to compose new ones keeping the description management this simple.

7 Final remarks

This paper has presented a flexible way for linking and describing I-TV programs and media objects, providing support for context awareness. We have revisited our previous model [20], which uses the best from two worlds: high level program descriptions from the TV-Anytime model and low level media descriptions from MPEG-7, with extensions to support our context framework and a complete separation of descriptions: media features, media composition, context information, linking and programs. This separation facilitates the reuse of descriptions and of pieces of descriptions during object composition, easing the development of appealing interactive applications such as those presented in Sect. 3.

Structured descriptions of media objects are made in the MediaObject schemas via selectively use of MPEG-7 standard. Using the MPEG-7 DDL, we have integrated in our model selected MPEG-7 descriptors and our context framework. The context framework aims allowing context descriptions, being a superset of our context library [20]. The elements of the library are imported by the MediaObject schema, allowing for descriptions with semantic information about objects such as video, video scenes, video frames and areas inside a frame. This is useful, for instance, during searches for related material, recommendation of relevant material, previewing content before access, and in context-aware systems that use this kind of information to make adaptations [9, 37, 38].

The context framework represents a contribution towards a comprehensive classification for context, which stills an open question. We have used a user context model in the framework, allowing description of any entity in that context. As described in Sect. 4, the framework can be used to describe other types of context, like systems or applications.

The underlying linking structure presented in our model, based on the XLink recommendation, is used to provide a mechanism that associates MPEG-4 media objects with their related descriptions (an open issue inside the MPEG committees). At the same time, this linking structure keeps the previous features [20] of both establishing links and describing relationships (spatial, temporal, directional, etc.) between descriptions.

As on going work, we are modeling system context in order to enhance the context framework. The system and user contexts can lead towards a future work where applications use context information to adapt content (I-TV programs) automatically, according to user's interests or system's capabilities. Also as a future work, we intent to automate the process of to acquire an MPEG-4 object and associate this ID with the object's MPEG-7 description. Another future work is to adapt our schemas to be MPEG-21 [39] compliant. The intention is to provide different users with different views of metadata and content at the same time.

Appendix A: Related technologies: MPEG-4 and MPEG-7

The MPEG-4 standard [1, 2] was developed to provide solutions to the new multimedia applications through features like: composition of presentations built from multimedia objects; streaming; error recovering; high compression and synchronization. The functionality of this standard is based on interactions with objects (video, audio, 2D and 3D graphics, 2D and 3D animations, text, etc.) present into a scene. A possible list of interactions types could be: to change objects' position and scale; to rotate objects; to change the speed; to add or to remove objects from the scene [1, 2].

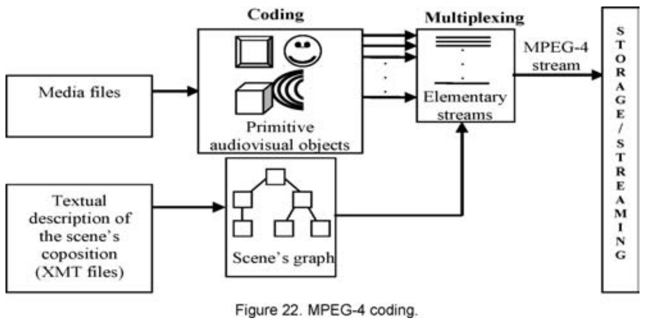

Coding
Multiplexing
Media files
Primitive audiovisual objects
Elementary streams
MPEG-4 stream
STORAGE/STREAMING
Textual description of the scene's coposition (XMT files)
Scene's graph
Figure 22. MPEG-4 coding.

Fig. 22 MPEG-4 coding

The authoring of interactive multimedia presentations is done arranging objects into a scene. To do that, the MPEG-4 species a language, called Extensible MPEG-4 Textual Format (XMT [2]), which has two versions: XMT-A and XMT-Ω. The XMT-A, based on VRML, represents a textual version of the MPEG-4 BIFS (Binary Format for Scenes [1, 2]). The XMT-Ω is a more readable (for authors) higher level approach, based on SMIL [13]. XMT allows authors to define objects, define and change objects' properties, place objects in space and time and associate actions with the objects. All these information, present in a XMT file, forms what the standard calls a scene's graph (Fig. 22). In order to produce a binary MPEG-4 presentation, both XMT-A and XMT-Ω must be translated to BIFS by the means of a MPEG-4 codec.

Figure 22 shows the coding, multiplexing and streaming processes of MPEG-4 objects: media files (like AVI videos) and a textual description of a scene composition are sent to the coder. The coder generates compressed primitive audiovisual MPEG-4 objects and outputs MPEG-4 elementary streams. These elementary streams are sent to the multiplexer, which outputs a single MPEG-4 stream holding the scene composition information and the elementary streams. The MPEG-4 stream can be stored in the file system or streamed over a network.

An interesting feature of the MPEG-4 standard is that one of those elementary streams can be a Java program – called MPEGLet [2]. The MPEGLets are decoded and executed at the client side. MPEGLets have access to the scene's graph and to the elementary streams, providing the developers with the means to build advanced interaction mechanisms with the MPEG-4 content being presented to the user.

The MPEG-7 standard [3] has the aim to standardize the description of data with multimedia content. This is done through the specification of a set of standard Descriptors, which can used to describe a variety of kinds of multimedia information. The MPEG-7 specifies predefined structures of descriptors and its relationships, and also specifies ways to the user build their own structures. These structures are called Description Schemes and the definition of new description schemes is made using a specific language called Description Definition Language – DDL [3, 4]. The DDL is based on the XML Schema language [30], containing structural and data type specifications of the XML Schema. However, the DDL adds some extensions related to the arrays and matrices data types, as well as some extensions related to description of time (duration and time point) [3, 25].

It is important to observe that the MPEG-7 standard do not aims to produce coding formats, searching mechanisms neither authoring tools. It is also important to observe that the MPEG-7 DLL is not a modeling language. The DDL is a language to represent the results of audiovisual data modeling [3, 4]. MPEG-7 descriptions are not limited to multimedia objects. Printed photos and documents, objects, animals and persons of the real world, or even abstractions like a TV program can be described in MPEG-7. Another important point is that MPEG-7 is independent of the others MPEG standards. It can used to describe objects in any format, like AVI, MOV, WAV, SMIL, etc.

References

1. ISO/IEC JTC1/SC29/WG11 N4668: Overview of the MPEG-4 Standard (2002): http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm
2. Pereira, F., Ebrahimi, T. (eds.): The MPEG-4 Book. Prentice Hall (2002)
3. ISO/IEC JTC1/SC29/WG11 N5525: MPEG-7 Overview (2003): http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.
4. Nack, F., Lindsay, A.T.: Everything you wanted to know about MPEG-7: Part 1. IEEE MultiMedia **6**(3), 65–77 (1999)
5. Merialdo, B., Lee, K.T., Luparello, D.: Automatic construction of personalized TV news programs. In: Proceedings of the ACM Multimedia, pp. 323–331 (1999)
6. Smyth, B., Wilson, D., O' Sullivan, D.: Improving the quality of the personalized electronic programme guide. In: Proceedings of the 2nd Workshop of Personalization in Future TV at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems, pp. 48–51 (2002)
7. Srivastava, H.O.: Interactive TV Technology and Markets. Artech House (2002)
8. Yu, B., Nahrstedt, K.: Internet-based interactive HDTV. Multimedia Syst. **9**, 477–489 (2004)
9. Abowd, G.D., Mynatt, E.D., Rodden, T.: The Human Experience. Pervasive Computing, pp. 48–57 (2002)
10. Dey, A.K.: Understanding and using context. Pers. Ubiq. Comp. J. **5**(1), 4–7 (2001)
11. Cosmas, E., Parker, Y., Schoonjas, P., Vaduva, A., Dosch, C., Schäfer, R., Erk, A., Mies, R., Bais, M., Schäfer, R., Stammnitz, P., Selinger, T., Klunsoyr, G., Pedersen, L., Bauer, S., Engelsberg, A., Klock, B., Evensen, G.: Custom TV with MPEG-4 and MPEG-7. In: Proceeding of the Colloquium on Interactive Television. IEE, Savoy-Place, London, UK (1999)
12. Santos Jr., J.B. dos, Goularte, R., Moreira, E.S., Faria, G.B.: The modeling of structured context-aware interactive environments. Trans. SDPS J. Integr. Des. Process Sci. **5**(4), 77–93 (2001)
13. World Wide Web Consortium (W3C). Synchronized Multimedia Integration Language (SMIL 2.0). http://www.w3.org/TR/smil20/
14. Macromedia, Inc.: Flash MX. http://www.macromedia.com/ software/flash/ (2003)
15. Apple Computers, Inc: QuickTime File Format. http://developer.apple.com/techpubs/quicktime/qtdevdocs/
16. Boll, S., Klas, W., Wandel, J.: A cross-media adaptation strategy for multimedia presentations. In: Proc. ACM Multimedia, pp. 37–46 (1999)
17. Chorianopoulos, K., Lekakos, G., Spinellis, D.: Intelligent user interfaces in the living room: Usability design for personalized television applications. In: Proc. ACM UIST, pp. 230–232 (2003)
18. Hu, M.J., Ye, J.: MD2L: Content description of multimedia documents for efficient process and search/retrieval. In: Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries, pp. 200–213 (1999)
19. Tsinaraki, C., Papadomanolakis, S., Christodoulakis, S.: Towards a two-layered video metadata model. In: Proceedings of the 12th International Workshop on Database and Expert Systems Applications, pp. 937–941 (2001)
20. Goularte, R., Moreira, E.D.S., Pimentel, M.G.C.: Structuring Interactive TV Documents. In: Proceedings of the ACM DocEng 2003, pp. 42–51 (2003)
21. World Wide Web Consortium (W3C): XML Linking Language (XLink) Version 1.0: http://www.w3.org/TR/xlink/
22. Benitez, A.B., Paek, S., Chang, S., Puri, A., Huang, Q., Smith, J.R., Li, C., Bergman, L.D., Judice, C.N.: Object-based multimedia content description schemes and applications for MPEG-7. Signal Process.: Image Communi. **16**(1,2), 235–269 (2000)
23. Dublin Core Metadata Initiative. http://dublincore.org
24. Lagoze, C., Hunter, J.: The ABC ontology and model (v3.0). J. Dig. Info. **2**(2) (2001) http://jodi.tamu.edu/Articles/v02/i02/Lagoze/
25. Salembier, P., Smith, J.R.: MPEG-7 multimedia description schemes. IEEE Trans. on Circuits Syst. Video Technol. **11**(6), 748–759 (2001)
26. Hjelsvold, R., Vdaygiri, S., Léauté, Y.: Web-based personalization and management of interactive video. In: Proceedings of the 10th International World Wide Web Conferece, pp. 129–139 (2001)
27. TV-Anytime Forum: Specification Series: S3: Metadata (Normative) (2001): ftp://tva:tva@ftp.bbc.co.uk/pub/Specifications/SP003v11.zip
28. Goularte, R., Cattelan, R.G., Camacho-Guerrero, J.A., Inácio, V.R. Jr., Pimentel, M.G.C.: Interactive multimedia annotations: Enriching and extending content. ACM DocEng 2004, pp. 84–86 (2004)
29. Goularte, R., Camacho-Guerrero, J.A., Inácio, V.R. Jr., Cattelan, R.G., Pimentel, M.G.C.: M4Note: a multimodal tool for multimedia annotation. In: Proceedings of the Latin-American Web Congress, pp. 142–149 (2004)
30. World Wide Web Consortium (W3C): XML Schema Part 0: Primer: http://www.w3.org/TR/xmlschema-0/
31. Tsinaraki, C., Fatourou, E., Christodoulakis, S.: An otology-driven framework for the management of semantic metadata describing audiovisual information. Lecture Notes on Computer Sciences 2681, pp. 340–356 (2003)
32. Zhu, X., Wu, X., Fan, J., Elmargamid, A.K., Aref, W.G.: Exploring video content structure for hierarchical summarization. Multi. Syst. **10**, 98–115 (2004)
33. World Wide Web Consortium (W3C): XML Pointer Language (XPointer) Version 1.0: http://www.w3.org/TR/xptr/
34. World Wide Web Consortium (W3C): Document Object Model (DOM) Level 3 Core Specification. http://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20020409/
35. Apache Software Foundation. Xerces Java Parser Readme. http://xml.apache.org/xerces-j/index.html
36. Fujitsu XLink Processor. http://www.labs.fujitsu.com/free/xlip/en/index.html
37. Harrison, B.L., Fishkin, K.P., Gujar, A., Mochon, C., Want R.: Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In: Proceedings ACM CHI 1998, pp. 17–24 (1998)
38. Truong, K.N., Abowd, G.D., Brotherton, J.A.: Who, what, when, where, how: design issues of capture & access applications. Lecture Notes on Computer Science 2201, Proc. Ubicomp, pp. 209–224 (2001)
39. ISO/IEC JTC1/SC29/WG11 N5231: MPEG-21 Overview (2001): http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm.
40. Ebrahimi, T., Horne, C.: MPEG-4 natural video coding – an overview. Signal Process. Image Commun. **15**(4,5), 365–385 (2000)