ELSEVIER

Signal Processing: *Image Communication* 15 (2000) 281–298

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

MPEG-4 Systems: Overview☆

Olivier Avaro[a,*], Alexandros Eleftheriadis[b], Carsten Herpel[c], Ganesh Rajan[d], Liam Ward[e]

[a]*Deutsche Telekom – Berkom GmbH, Abt. T.22, D-64307 Darmstadt, Deutschland, Germany*
[b]*Columbia University, Dept. of Electrical Engineering, 500 West 120th Street, Mail Code 4712, New York, NY 10027, USA*
[c]*Deutsche Thomson-Brandt GmbH, Karl-Wichert-Allee 74, 30625 Hannover, Deutschland, Germany*
[d]*General Instrument, 6450 Sequence Dr., San Diego, 92121 CA, USA*
[e]*Teltec Ireland, DCU, Dublin 9, Ireland*

Abstract

This paper gives an overview of Part 1 of ISO/IEC 14496 (MPEG-4 Systems). It first presents the objectives of the MPEG-4 activity. In the MPEG-1 and MPEG-2 standards, "Systems" referred only to overall architecture, multiplexing, and synchronization. In MPEG-4, in addition to these issues, the Systems part encompasses scene description, interactivity, content description, and programmability. The description of the MPEG-4 specification follows, starting from the general architecture up to the description of the individual MPEG-4 Systems tools. Finally, a conclusion describes the future extensions of the specification, as well as a comparison between the solutions provided by MPEG-4 Systems and some alternative technologies. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: APIs; Architecture; Audio-visual; Buffer management; Composition; Content description; Interactivity; MPEG-4 systems; Multiplex; Programmability; Scene description; Specification; Synchronization; Tools

1. Introduction

The concept of "Systems" in MPEG has evolved dramatically since the development of the MPEG-1 and MPEG-2 standards. In the past, "Systems" referred only to overall architecture, multiplexing, and synchronization. In MPEG-4, in addition to these issues, the Systems part encompasses scene description, interactivity, content description, and programmability. The combination of the exciting new ways of creating compelling interactive audio-visual content offered by MPEG-4 Systems, and the efficient representation tools provided by the Visual and Audio parts, promise to be the foundation of a new way of thinking about audio-visual information.

This paper gives an overview of MPEG-4 Systems. It is structured around the objectives, architecture, and the tools of MPEG-4 Systems as follows:

- *Objectives*: This section describes the motivations and the rationale behind the development of the MPEG-4 Systems specifications. As with

* Corresponding author.

☆ This paper is a revised text from the original publication Advances in Multimedia: Systems, Standards, and Networks, Chapter 12 by Olivier Avaro, Editors: Atul Puri and Tsuhan Chen, originally published by Marcel Dekker Inc.

E-mail addresses: o.avaro@berkom.de (O. Avaro), eleft@ee.columbia.edu (A. Eleftheriadis), herpelc@thmulti.com (C. Herpel), ganesh_rajan@hotmail.com (G. Rajan), liam.ward@teltec.dcu.ie (L. Ward)

all MPEG activities, MPEG-4 Systems is guided by a set of requirements [8], i.e., the set of objectives that must be satisfied by the specifications resulting from the work or activities of the sub-group. This paper give a particular attention to the way the requirements of MPEG-4 Systems are derived from the principal concept behind MPEG-4, viz., the coding of audio-visual objects.
- *Architecture*: This section describes the overall structure of MPEG-4, known as the "MPEG-4 Systems Architecture". A complete walkthrough of an MPEG-4 session highlights the different phases that a user will, in general, follow in consuming MPEG-4 content.
- *Tools*: MPEG-4 is a "toolbox" standard, providing a number of tools, sets of which are particularly suited to certain applications. This section provides a functional description of the MPEG-4 Systems tools. These tools are further described in the sections that follow, and are fully specified in [3,9].

Of course, MPEG-4 is not the only initiative that attempts to provide solutions in the area described above. Several companies, industry consortia, and even other standardization bodies have developed technologies that, to some extent, also aim to address objectives similar to those of MPEG-4 Systems. In concluding this look at MPEG-4 Systems, this paper provides an overview of some of these alternative technologies and makes a comparison with the solutions provided by MPEG-4 Systems.

2. Objectives

2.1. Requirements

To understand the rationale behind the activity, a good starting point is one of the most fundamental MPEG-4 documents, viz., the MPEG-4 requirements [8]. This document gives an extensive list of the objectives that needed to be satisfied by the MPEG-4 specifications. The goal of specifying a standard way for the description and coding of audio-visual objects was the primary motivation behind the development of the tools in the MPEG-4 Systems.

MPEG-4 Systems requirements may be categorized into two groups:
- *Traditional MPEG Systems requirements*: The core requirements for the development of the systems specifications in MPEG-1 and MPEG-2 were to enable the transport of coded audio, video and user-defined private data, and to incorporate timing mechanisms to facilitate synchronous decoding and presentation of these data at the client side. These requirements also constitute a part of the fundamental requirements set for MPEG-4 Systems. The evolution of the traditional MPEG Systems activities to match the objectives for MPEG-4 Systems is detailed in Section 2.2.
- *Specific MPEG-4 Systems requirements*: The requirements in this set, most notably, the notions of *audio-visual objects* and *scene description*, represent the ideas central to MPEG-4 and are completely new in MPEG Systems. The core competencies needed to fulfil these requirements were not present at the beginning of the activity but were acquired during the standards development process. Section 2.3 describes these specific MPEG-4 Systems requirements.

To round out this discussion on the MPEG-4 objectives, Section 2.4 finally provides an answer to the question "What is MPEG-4 Systems?" by summarizing the objectives of the MPEG-4 Systems activity and describing the charter of the MPEG-4 Systems sub-group during its four years of existence.

2.2. Traditional MPEG Systems requirements

The work of MPEG traditionally addressed the representation of audio-visual information. In the past, this included only natural audio and video material.[1] As we will indicate in subsequent sections, the types of media included within the scope of the MPEG-4 standards have been significantly extended. Regardless of the type of the media, each

[1] "Natural", in this context, is generally understood to mean representations of the real-world that are captured using cameras, microphones and so on, as opposed to synthetically generated material.

one has spatial and/or temporal attributes and needs to be identified and accessed by the application consuming the content. This results in a set of requirements for MPEG Systems on streaming, synchronization and stream management, further described below.

- *Streaming*: The audio-visual information is to be delivered in a streaming manner, suitable for live broadcast of such content. In other words, the audio-visual data are to be transmitted piece by piece, in order to match the delivery of the content to clients with limited network and terminal capabilities. This is in stark contrast to some of the existing scenarios, the World Wide Web, for example, wherein the audio-visual information is completely downloaded on to the client terminal and then played back. It was thought that such scenarios would necessitate too much storage on the client terminals for applications envisaged by MPEG.
- *Synchronization*: Typically, the different components of an audio-visual presentation are closely related in time. For most applications, audio samples with associated video frames have to be presented together to the user at precise instants in time. The MPEG representation needs to allow a precise definition of the notion of time so that data received in a streaming manner can be processed and presented at the right instants in time, and be temporally synchronized with each other.
- *Stream Management*: Finally, the complete management of streams of audio-visual information implies the need for certain mechanisms to allow an application to consume the content. These include mechanisms for unambiguous location of the content, identification of the content type, description of the dependencies between content elements, access to the intellectual property information associated to the data, etc.

In the previous MPEG-1 and MPEG-2 standards, these requirements led to the definition of the following tools:

1. *Systems target decoder* (*STD*): The Systems Target Decoder is an abstract model of an MPEG decoding terminal that describes the idealized decoder architecture and defines the behavior of its architectural elements. The STD provides for a precise definition of time and recovery of timing information from information encoded within the streams themselves, as well as mechanisms to synchronize streams with each other. It also allows for the management of the decoder's buffers.
2. *Packetization of streams*: This set of tools defines the organization of the various audio-visual data into streams. First, definition of the structure of individual streams containing data of a single type is provided (e.g., a video stream), followed by the multiplexing of different individual streams for transport over a network or storage in disk files. At each level, additional information is included to allow for the complete management of the streams (synchronization, intellectual property rights, etc.).

All these requirements are still relevant for MPEG-4. However, the existing tools needed to be extended and adapted for the MPEG-4 context. In some cases, these requirements led to the creation of new tools. More specifically:

1. *Systems decoder model* (*SDM*): The nature of MPEG-4 streams can be different from the ones dealt with in the traditional MPEG-1 and MPEG-2 decoder models. For example, MPEG-4 streams may have a bursty data delivery schedule. They may be downloaded and cached before their actual presentation to the user. Moreover, to implement the MPEG-4 principle of "create once, access everywhere", the transport of content does not need to be (indeed, should not be) integrated into the overall architecture. These new aspects, therefore, led to a modification of the MPEG-1 and MPEG-2 models that resulted in the *MPEG-4 Systems Decoder Model*.
2. *Synchronization*: The MPEG-4 principle of "create once, access everywhere" is easier to achieve when all of the content-related information forms part of the encoded representation of the multimedia content. This content-related information includes synchronization information also. The observation that the range of bit rates addressed by MPEG-4 is broader than in

MPEG-1 and MPEG-2, and can be from a few kbit/s up to several Mbit/s led to the definition of a flexible tool to encode the synchronization information, the *Sync Layer* (*Synchronization Layer*).

3. *Packetization of streams*: On the delivery side, most of the existing networks provide ways for packetization and transport of streams. Therefore, beyond defining the modes for the transport of MPEG-4 content on the existing infrastructures, MPEG-4 Systems did not see a need to develop any new tools for this purpose. However, due to the possibly unpredictable temporal behavior of MPEG-4 data streams as well as the possibly large number of such streams in MPEG-4 applications, MPEG-4 Systems developed an efficient and simple multiplexing tool to enhance the transport of MPEG-4 data: the *FlexMux* (*Flexible Multiplex*) tool.

2.3. MPEG-4 Specific Systems requirements

The foundation of MPEG-4 is the coding of *audio-visual objects*. As per MPEG-4 terminology, an audio-visual object is the representation of a natural or synthetic object that has an audio and/or visual manifestation. Examples of audio-visual objects include a video sequence (perhaps with shape information), an audio track, an animated 3D face, speech synthesized from text, or a background consisting of a still image.

The advantages of coding audio-visual objects can be summarized as follows:

- It allows interaction with the content. At the client side, users can be given the possibility to access, manipulate, or activate specific parts of the content.
- It improves reusability and coding of the content. At the content creation side, authors can easily organize and manipulate individual components and reuse existing material. Moreover, each type of content can be coded using the most effective algorithms. Artifacts due to joint coding of heterogeneous objects (e.g., graphics overlaid on natural video) disappear.
- It allows content-based scalability. At various stages in the authoring/delivery/consumption process, content can be ignored or adapted to match bandwidth, complexity, or price requirements.

In order to be able to use these audio-visual objects in a presentation, additional information needs to be transmitted to the client terminals. The individual audio-visual objects are only a part of the presentation structure that an author wants delivered to the consumers. Indeed, for the presentation at the client terminals, the coding of audio-visual objects needs to be augmented by the following:

1. The coding of information that describes the spatio-temporal relationships between the various audio-visual objects present in the presentation content. In MPEG-4 terminology, this information is referred to as the *Scene description* information.
2. The coding of information that describes how time-dependent objects in the scene description are linked to the streamed resources actually transporting the time-dependent information.

These considerations imply additional requirements for the overall architectural design, which are summarized below:

- *Object description*: In addition to the identification of the location of the streams, other information may need to be attached to streamed resources. This may include the identification of streams in alternative formats, a scalable stream hierarchy that may be attached to the object or description of the coding format of the object.
- *Content authoring*: The object description information is conceptually different from the scene description information. It will therefore have different life cycles. For example, at some instant in time, object description information may change (like the intellectual property rights of a stream or the availability of new streams) while the scene description information remains the same. Similarly, the structure of the scene may change (like changing the positions of the objects), while the streaming resources remain the same.
- *Content consumption*: The consumer of the content may wish to obtain information *about* the

content (e.g., the intellectual property attached to it or the maximum bit-rate needed to access it) before actually requesting it. He then only needs to receive information about object description, not about the scene description.

Besides the coding of audio-visual objects organized spatio-temporally, according to a scene description, one of the key concepts of MPEG-4 is the idea of interactivity, that is, that the content reacts upon the action of a user. This general idea is expressed in three specific requirements:

1. *Client side interaction*: The user should be able to manipulate the scene description as well as the properties of the audio-visual objects that the author wants to expose to interaction.
2. *Audio-visual objects behavior*: It should be possible to attach behavior to audio-visual objects. User actions or other events, like time, trigger these behaviors.
3. *Client–Server interaction*: Finally, in case a return channel from the client to the server is available, the user should be able to send back information to the server that will act upon it and eventually send updates or modification of the content.

2.4. *What is MPEG-4 Systems*?

The main concepts that were described in this section are depicted in Fig. 1. The mission, therefore, of the MPEG-4 Systems activity may be

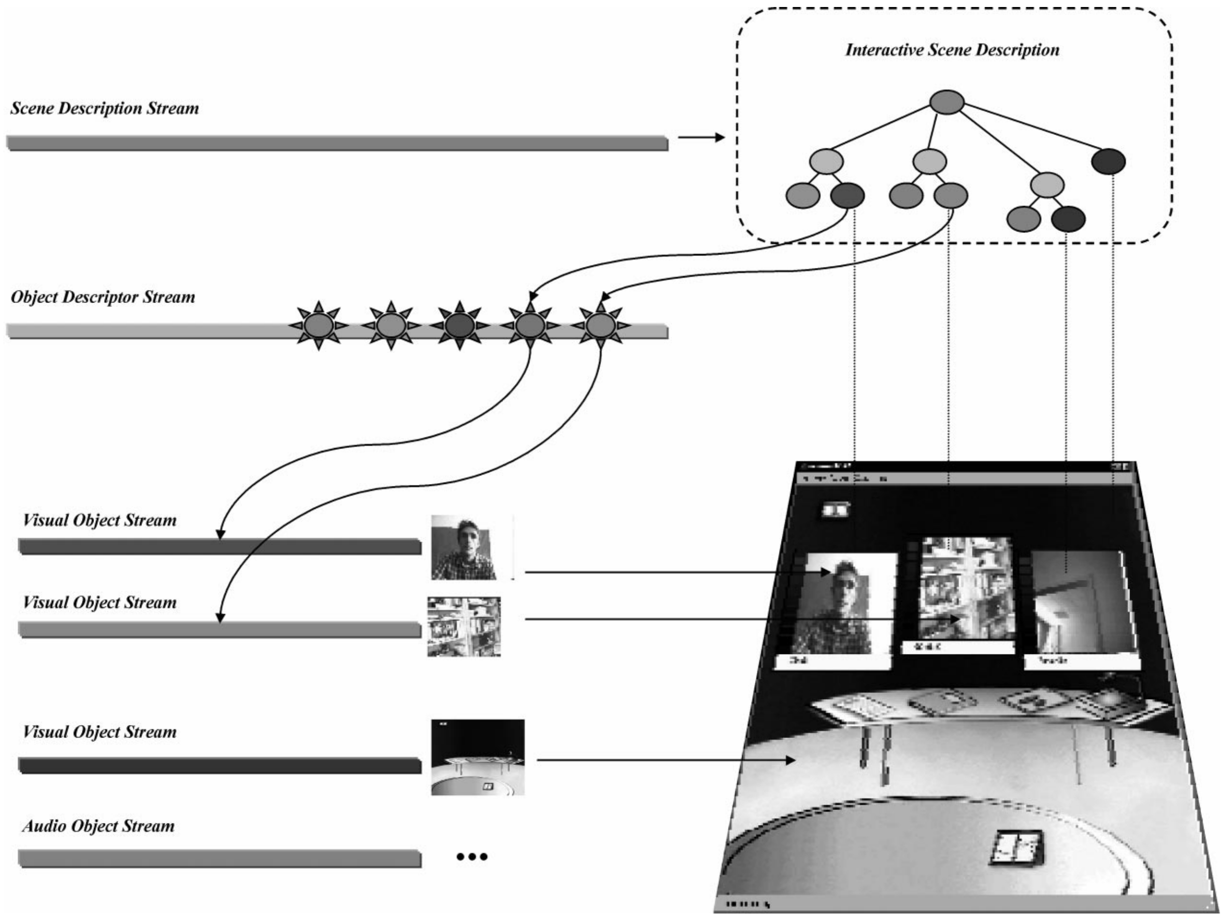

Fig. 1. MPEG-4 systems principles.

summarized by the following sentence: *"Develop a coded, streamable representation for audio-visual objects and their associated time-variant data along with a description of how they are combined"*.

More precisely, in this sentence:

- "*Coded representation*" should be seen in contrast to "*textual representation*". Indeed, all the information that MPEG-4 Systems contains (scene description, object description, synchronization information) is binary encoded for bandwidth efficiency.
- "*Streamable*" should *not* be seen in contrast to "*stored*", since storage and transport are dealt with in a similar and consistent way in the MPEG-4 framework. It should rather be seen in contrast to "*downloaded*". Indeed, MPEG-4 is built on the concept of streams that have a temporal extension, and not on the concept of files of finite size.
- "*Elementary audio-visual sources along with a description of how they are combined*" should be seen in contrast to "*individual audio or visual streams*". MPEG-4 System does not deal with the encoding of audio or visual information but only with the information related to the combinations of streams: combination of audio-visual objects to create an interactive audio-visual scene, synchronization of streams, multiplexing of streams for storage or transport. The term "description" here reinforces the fact that the combination involves explicit content authoring.

3. Architecture

The overall architecture of an MPEG-4 terminal is depicted in Fig. 2. Starting at the bottom of the figure, we first encounter the particular storage or transmission medium. This refers to the lower layers of the delivery infrastructure (network layer and below, as well as storage). The transport of the MPEG-4 data can occur on a variety of delivery systems. This includes MPEG-2 Transport Streams, UDP (User Datagram Protocol) over IP (Internet Protocol), ATM (asynchronous transfer mode) AAL2 (ATM adaptation layer 2), MPEG-4 (MP4) files or the DAB (digital audio broadcasting) multiplexer.

Most of the currently available transport layer systems provide native means for multiplexing information. There are, however, a few instances where this is not the case, like in GSM (Global Systems for Mobile communication) data channels. In addition, the existing multiplexing mechanisms may not fit MPEG-4 needs in terms of low delay, or they may incur substantial overhead in handling the expected large number of streams associated with an MPEG-4 session. As a result, the FlexMux tool can optionally be used on top of the existing transport delivery layer.

Regardless of the transport layer used and the use (or not) of the FlexMux option, the delivery layer provides to the MPEG-4 terminal a number of elementary streams. Note that not all of the streams have to be downstream (server to the client); in other words, it is possible to define elementary streams for the purpose of conveying data back from the terminal to the transmitter or server.

In order to isolate the design of MPEG-4 from the specifics of the various delivery systems, the concept of the DMIF (delivery multimedia integration framework) application interface (DAI) [7] was defined. This interface defines the process of exchanging information between the terminal and the delivery layer in a conceptual way, using a number of primitives. It should be pointed out that this interface is non-normative; MPEG-4 terminal implementations do not need to expose such interface.

The DAI defines procedures for initializing an MPEG-4 session and obtaining access to the various elementary streams that are contained in it. These streams can contain a number of different informations: audio-visual object data, scene description information, control information in the form of object descriptors, as well as meta-information that describes the content or associates intellectual property rights to it.

Regardless of the type of data conveyed in each elementary stream, it is important that they use a common mechanism for conveying timing and framing information. The Sync Layer (SL) is defined for this purpose. It is a flexible and configurable packetization facility that allows the inclusion

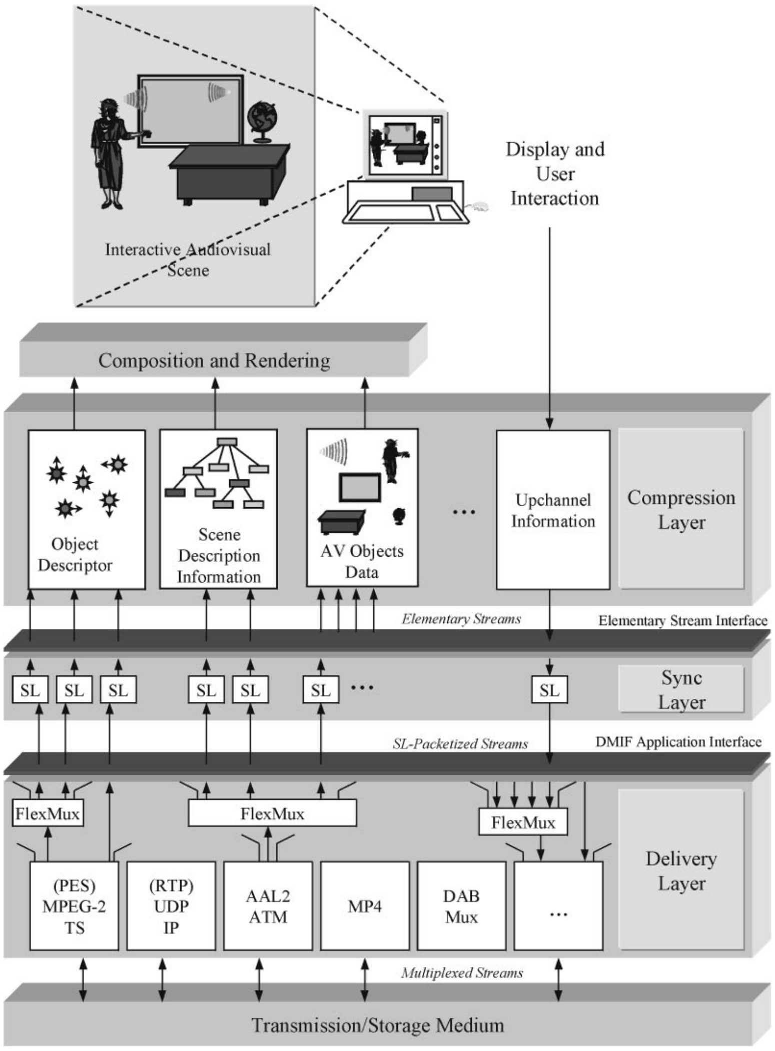

Fig. 2. MPEG-4 systems architecture.

of timing, fragmentation, and continuity information on associated data packets. Such information is attached to data units that comprise complete presentation units, e.g., an entire video object plane (VOP) or an audio frame. These are called *access units*. An important feature of the SL is that it does not contain frame demarcation information; in other words, the SL header contains no packet length indication. This is because it is assumed that the delivery layer that processes SL packets will already make such information available. Its exclusion from the SL thus eliminates duplication.

The SL is the sole mechanism of implementing timing and synchronization mechanisms in MPEG-4. The fact that it is highly configurable allows the use of several different models. At one end of the spectrum, traditional clock recovery methods using clock references and time stamps can be used. It is also possible to use a rate-based approach (rather than using explicit timestamps, the known rate of the access units implicitly determines their time stamps). At the other end of the spectrum, it is possible to operate without any clock information – data are processed as soon as they arrive. This would be suitable, for example, for a slide-show presentation. The primary mode of operation, and the one supported by the currently defined conformance points of the specification, involves the full complement of clock recovery and time stamps. By defining a system decoder model, this makes it possible to both synchronize the receiver's clock to the sender's, as well as manage the buffer resources at the receiver.

From the SL information we can recover a time base as well elementary streams. The streams are sent to their respective decoders that process the data and produce composition units (e.g., a decoded video object plane). In order for the receiver to know what type of information is contained in each stream, control information in the form of *object descriptors* is used. These descriptors associate sets of elementary streams to one audio or visual object, define a scene description stream, or even point to an object descriptor stream. These descriptors, in other words, are the way with which a terminal can identify the content being delivered to it. Unless a stream is described in at least one object descriptor, it is impossible for the terminal to make use of it.

At least one of the streams must be the scene description information associated with the content. The scene description information defines the spatial and temporal position of the various objects, their dynamic behavior, as well as any interactivity features made available to the user. As mentioned above, the audio-visual object data is actually carried in its own elementary streams. The scene description contains *pointers* to object descriptors when it refers to a particular audio-visual object. We should stress that it is possible that an object (in particular synthetic objects like text and simple graphics) may be fully described by the scene description. As a result, it may not be possible to uniquely associate an audio-visual object with just one syntactic component of MPEG-4 Systems. As detailed in Section 4.2, the scene description is tree-structured and is heavily based on VRML (Virtual Reality Modeling Language [4]) structure.

A key feature of the scene description is that, since it is carried in its own elementary stream(s), it can contain full timing information. This implies that the scene can be dynamically updated over time, a feature that provides considerable power for content creators. In fact, the scene description tools provided by MPEG-4 also provide a special lightweight mechanism to modify parts of the scene description in order to effect animation. This is accomplished by coding, in a separate stream, only the parameters that need to be updated.

The system's *compositor* uses the scene description information, together with decoded audio-visual object data, in order to render the final scene that is presented to the user. It is important to note that the MPEG-4 Systems architecture does not define how information is to be rendered. In other words, the Systems part of the MPEG-4 standard does not detail mechanisms through which the values of the pixels to be displayed or audio samples to be played back can be uniquely determined. This is an unfortunate side-effect of providing synthetic content representation tools. Indeed, in the general case, it is not possible to define rendering without venturing into issues of terminal implementation. Although this makes compliance testing much more difficult (requiring subjective evaluation), it allows the inclusion of a very rich set of synthetic content representation tools. In some cases however, like in the Audio part of the MPEG-4 standard [6], composition can be and is fully defined.

The scene description tools provide mechanisms to capture user or system events. In particular, they allow the association of events to user operations on desired objects, that can – in turn – modify the behavior of the stream. Event processing is the core mechanism with which application functionality and differentiation can be provided. In order to provide flexibility in this respect, MPEG-4 allows the use of ECMAScript (also known as JavaScript) scripts within the scene description. Use of scripting tools is essential in order to access state information and implement sophisticated interactive applications.

It is important to point out that, in addition to the new functionalities that MPEG-4 makes available to content consumers, it provides tremendous advantages to content creators as well. The use of an object-based structure, where composition is performed at the receiver, considerably simplifies the content creation process. Starting from a set of coded audio-visual objects, it is very easy to define a scene description that combines these objects in a meaningful presentation. A similar approach is essentially used in HTML (Hyper Text Markup Language) and Web browsers, thus allowing even non-expert users to easily create their own content. The fact that the content's structure survives the process of coding and distribution, also allows for its reuse. For example, content filtering and/or searching applications can be easily implemented using ancillary information carried in object descriptors (or its own elementary streams, as described in Section 4.1). Also, users themselves can easily extract individual objects, assuming that the intellectual property information allows them to do so.

In the following section the different components of this architecture are described in more detail.

4. Tools

4.1. *Stream management: the object description framework*

The object description framework provides the glue between the scene description and the streaming resources – the elementary streams – of an MPEG-4 presentation, as indicated in Fig. 1. Unique identifiers are used in the scene description to point to the *object descriptor*, the core element of the object description framework. The object descriptor is a container structure that encapsulates all of the setup and association information for a set of elementary streams. A set of sub-descriptors, contained in the object descriptor, describe the individual elementary streams, including the configuration information for the stream decoder as well as the flexible sync layer syntax for this stream. Each object descriptor, in turn, groups a set of streams that are seen as a single entity from the perspective of the scene description.

Object descriptors are transported in dedicated elementary streams, called object descriptor streams, that make it possible to associate timing information to a set of object descriptors. With the appropriate wrapper structures, called *OD commands* (*object descriptor commands*), around each object descriptor, it is possible to update and remove each object descriptor in a dynamic and timely manner. The existence or the absence of descriptors determines the availability (or the lack thereof) of the associated elementary streams to the MPEG-4 terminal.

The *initial object descriptor*, a derivative of the object descriptor, is a key element necessary for accessing the MPEG-4 content. It conveys content complexity information in addition to the regular elements of an object descriptor. As depicted in Fig. 3, the initial object descriptor usually contains at least two elementary stream descriptors. One of the descriptor must point to a scene description stream while the others may point to an object descriptor stream. This object descriptor stream transports the object descriptors for the elementary streams that are referred to by some of the components in the scene description. Initial object descriptors may themselves be transported in object descriptor streams since they allow content to be hierarchically nested, but may as well be conveyed by other means, serving as starting pointers to MPEG-4 content.

In addition to providing essential information about the relation between the scene description and the elementary streams, the object description framework provides mechanisms to describe hierarchical relations between streams, reflecting scalable encoding of the content and means to indicate multiple alternate representations of content. Furthermore, textual descriptors about content items, called *object content information* (OCI), and descriptors for the intellectual property rights management and protection (IPMP) have been defined. The latter allow conditional access or other content control mechanisms to be associated to a particular content item. These mechanisms may be different on a stream-by-stream basis and possibly even a multiplicity of such mechanisms could co-exist.

A single MPEG-4 presentation, or program, may consist of a large number of elementary streams with a multiplicity of data types. The object description framework has been separated from the scene description to account for this fact and the related consequence that service providers may possibly wish to relocate streams in a simple way.

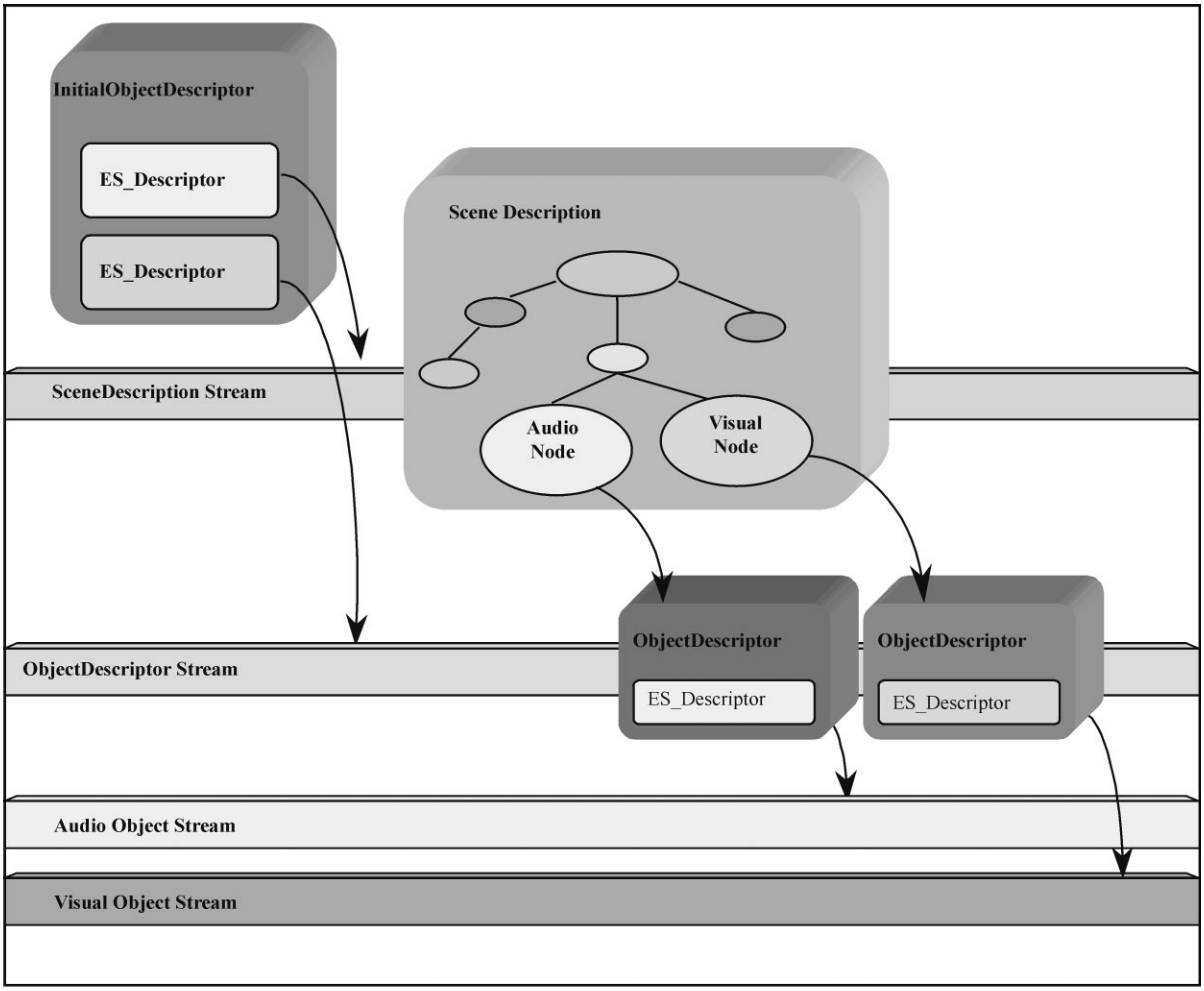

Fig. 3. The initial object descriptor and the linking of elementary streams to the scene description.

Such relocation may require changes in the object descriptors; however, it will not affect the scene description. Therefore, object descriptors improve content manageability.

The reader is referred to the MPEG-4 Systems specification [3] for the syntax and semantics of the various components of this framework and their usage within the context of MPEG-4.

4.2. Presentation engine: BIFS

MPEG-4 specifies a binary format for scenes (BIFS) that is used to describe scene composition information: the spatial and temporal locations of objects in scenes, along with their attributes and behaviors. Elements of the scene and the relationships between them form the scene graph that must be coded for transmission. The fundamental scene graph elements are the "nodes" that describe audio-visual primitives and their attributes, along with the structure of the scene graph itself. BIFS draws heavily on this and other concepts employed by VRML[2] [4].

[2] VRML was proposed and developed by the VRML Consortium and their VRML 2.0 specification became an ISO/IEC International Standard in 1998.

Designed as a file format for describing 3D models and scenes ("worlds" in VRML terminology), VRML lacks some important features that are required for the types of multimedia applications targeted by MPEG-4. In particular, the support for natural video and audio are basic (ex: streaming of audio or video objects are not supported) and the timing model is loosely specified, implying that synchronization in a scene consisting of multiple media types cannot be guaranteed. Furthermore, VRML worlds are often very large (ex: there is neither compression nor animations streaming). Animations lasting around 30 s typically consume several megabytes of disk space. The strength of VRML is its scene graph description capabilities and this strength has been the basis upon which MPEG-4 scene description has been built.

BIFS includes support for almost all of the nodes in the VRML specifications. In fact, BIFS is essentially a superset of VRML, although there are some exceptions. BIFS does not yet support the PROTO and EXTERNPROTO nodes, nor does it support the use of Java language in the *Script* nodes (BIFS only supports ECMAScript). BIFS does, however, expand significantly on VRML's capabilities in ways that allow a much broader range of applications to be supported. Note that a fundamental difference between the two is that BIFS is a *binary* format, whereas VRML is a *textual* format. So, although it is possible to design scenes that are compatible with both BIFS and VRML, transcoding of the representation formats are required.

Here, we highlight the functionalities that BIFS adds to the basic VRML set. Readers unfamiliar with VRML might find it useful to first acquire some background knowledge from [4].

1. *Compressed binary format*: BIFS describes an efficient binary representation of the scene graph information. The coding may be either lossless or lossy. The coding efficiency derives from a number of classical compression techniques, plus some novel ones. The knowledge of context is exploited heavily in BIFS. This technique is based on the fact that given some scene graph data have been previously received, it is possible to anticipate the type and format of data to be received subsequently. This technique is described in Fig. 4. Quantization of numerical values is supported, as well as the compression of 2D meshes as specified in MPEG-4 Visual [5].

Consider a simple coding scheme with the following tags:

`<begin>`	- beginning of record
`<end>`	- end of record
`<break>`	- end of element in record
`<string>`	- text string follows
`<number>`	- number follows

We wish to use this scheme code a record consisting of first name, last name and phone number, for example:

First name:	Jim
Last name:	Brown
Phone:	777 1234

With no knowledge context we would need to code this as:

```
<start><string>Jim<break><string>Brown<break><number>7771234<break><end>
```

If the context is known, i.e. we know that the structure of the record is "string, string, number" we do not have to spend bits specifying the type of each element:

```
<start>Jim<break>Brown<break>7771234<end>
```

Fig. 4. Simple example of the use of context in efficient coding.

2. *Streaming*: BIFS is designed so that the scene may be transmitted as an initial scene followed by timestamped modifications to the scene. For dynamic scenes that change over time, this leads to a huge improvement in memory usage and reduced latency when compared to equivalent VRML scenes. The *BIFS Command* protocol allows replacement of the entire scenes, addition/deletion/replacement of nodes and behavioral elements in the scene graph as well as modification of scene properties.
3. *Animation*: A second streaming protocol, *BIFS Anim*, is designed to provide a low-overhead mechanism for the continuous animation of changes to numerical values of the components in the scene. These streamed animations provide an alternative to the interpolator nodes supported in both BIFS and VRML. The main difference is that interpolator nodes typically contain very large amounts of data that must be loaded in the scene and stored in memory. By streaming these animations, the amount of data that must be held in memory is reduced significantly. Secondly, by removing these data from the scene graph that must be initially loaded, the amount of data that must be processed (and therefore the time taken) in order to begin presenting the scene is also reduced.
4. *2D Primitives*: BIFS includes native support for 2D scenes. This facilitates content creators who wish to produce low-complexity scenes, including the traditional television and multimedia industries. Many applications cannot bear the cost of requiring decoders to have full 3D rendering and navigation. This is particularly true where hardware decoders must be of low cost, as for instance television set-top boxes. However, rather than simply partitioning the multimedia world into 2D and 3D, MPEG-4 BIFS allows the combination of 2D and 3D elements in a single scene.
5. *Enhanced audio*: VRML provides simple audio support. This support has been enhanced in MPEG-4. BIFS provides the notion of an "audio scene graph" where the audio sources, including streaming ones, can be mixed. It also provides nodes interface to the various MPEG-4 audio objects [6]. Audio content can even be processed and transformed with special procedural code to produce various sounds effects [6].
6. *Facial animation*: BIFS provides support at the scene level for the MPEG-4 facial animation decoder [5]. A special set of BIFS nodes expose the properties of the animated face at the scene level, making it a full participant of the scene that can be integrated with all BIFS functionalities, similarly to any other audio or visual objects.

4.3. *Timing and synchronization: the systems decoder model (SDM) and the sync layer*

The MPEG-4 SDM is conceived as an adaptation of its MPEG-2 predecessor. The System Target Decoder in MPEG-2 is a model that precisely describes the temporal and buffer constraints under which a set of elementary streams may be packetized and multiplexed. Due to the generic approach taken towards stream delivery – which includes stream multiplexing – MPEG-4 chose not to define multiplexing constraints in the SDM. Instead, the SDM assumes the concurrent delivery of an arbitrary number of – already demultiplexed – elementary streams to the decoding buffers of their respective decoders. A constant end-to-end delay is assumed between the encoder output and the input to the decoding buffer on the receiver side. This leaves the task of handing the delivery jitter (including multiplexing) to the delivery layer.

Timing of streams is expressed in terms of decoding and composition time of individual access units within the stream. Access units are the smallest sets of data to which individual presentation time stamps can be assigned (e.g., a video object plane). The decoding time stamp indicates the point in time at which an access unit is removed from the decoding buffer, instantaneously decoded, and moved to the composition memory. The composition time stamp allows the separation of decoding and composition times, to be used for example in the case of bi-directional prediction in visual streams. This idealized model allows the encoding side to monitor the space available in the decoding side's buffers, thus helping it, for example, to schedule ahead-of-time delivery of data. Of course, a resource management for the memory for decoded data would be desirable as well. However, it was acknowledged that this issue is strongly linked with memory use for the composition process itself, which is considered outside the scope of MPEG-4. Therefore, management of composition buffers is not part of the model.

Time stamps are readings of an object time base (OTB) that is valid for an individual stream or a set of elementary streams. At least all the streams belonging to one audio-visual object have to follow the same OTB. Since the OTB, in general, is not a universal clock, object clock reference time stamps can be conveyed periodically with an elementary stream to make it known to the receiver. This is, in fact, done on the wrapper layer around elementary streams, called the sync layer (SL). The sync layer provides the syntactic elements to encode the partitioning of elementary streams into access units and to attach both decoding and composition time stamps as well as object clock references to a stream. The resulting stream is called an SL-packetized stream. This syntax provides a uniform shell around elementary streams, providing the information that needs to be shared between the compression layer and the delivery layer in order to guarantee timely delivery of each access unit of an elementary stream.

Different from its predecessor in MPEG-2, the packetized elementary stream (PES), the sync layer does not constitute a self-contained stream, but rather a packet-based interface to the delivery layer. This takes into account the properties of typical delivery layers like IP, H.223 or MPEG-2 itself, into which SL-packetized streams are supposed to be mapped. There is no need to encode either unique start codes or the length of an SL packet within the packet, since synchronization and length encoding are already provided by the mentioned delivery layer protocols.

Furthermore, MPEG-4 has to operate both at very low and rather high bit-rates. This has led to a flexible design of the sync layer elements, making it possible to encode time stamps of configurable size and resolution, as required in a specific content or application scenario. The flexibility is made possible by means of a descriptor that is conveyed as part of the elementary stream descriptor that summarizes the properties of each (SL-packetized) elementary stream.

4.4. The transport of MPEG-4 content

Delivery of MPEG-4 content is a task that is supposed to be dealt with outside the MPEG-4 systems specification. All access to delivery layer functionality is conceptually done only through a semantic interface, called the DMIF application interface (DAI). It is specified in Part 6 of MPEG-4, Delivery multimedia integration framework (DMIF) [7]. In practical terms this means that the specification of control and data mapping to underlying transport protocols or storage architectures is to be done jointly with the respective organization that manages the specification of the particular delivery layer. For example, for the case of MPEG-4 transport over IP, development work is done jointly with the Internet Engineering Task Force (IETF).

An analysis of existing delivery layer properties showed that there might be a need for an additional layer of multiplexing, in order to map the occasionally bursty and low bit-rate MPEG-4 streams to a delivery layer protocol that exhibits fixed packet size or too much packet overhead. Furthermore, the provision of a large number of delivery channels may have a substantial burden in terms of management and cost. Therefore, a very simple multiplex packet syntax has been defined, called the FlexMux. It allows multiplexing a number of SL-packetized streams into a self-contained FlexMux stream with rather low overhead. It is proposed as an option to designers of the delivery layer mappings but is not used for the definition of MPEG-4 conformance points.

5. Conclusion

5.1. Extensions of the specification

The technologies considered for standardization in MPEG-4 were not all identically mature. Therefore, the MPEG-4 project in general and MPEG-4 Systems in particular, was organized in two phases:

Version 1 and Version 2. The tools described above already contain the majority of the functionality of MPEG-4 Systems and allow the development of compelling multimedia applications. These are provided by the current MPEG-4 standard, the so-called MPEG-4 "Version 1". Extension of the standard in the form of amendments, the so-called "Version 2", completes the Version 1 toolbox with new tools and new functionalities. Version 2 tools are not intended to replace any of the Version 1 tools. On the contrary, Version 2 is a completely backward compatible extension of Version 1.

Version 2 will provide for the following additional BIFS functionalities:

- Support of all the VRML constructs that were not supported in Version 1. The major ones are PROTO and EXTERNPROTO, allowing a more efficient compression of the scene description, a more flexible content authoring as well as a robust mechanism to define BIFS extensions;
- Specification of BIFS interfaces for new Version 2 audio and visual media such as synthetic body description and animation or 3D model coding;
- Specification of advanced audio BIFS for more natural sound source and sound environment modeling. These new audio functionalities include modeling of air absorption, of the audio response of the visual scene, and more natural modeling of distance dependent attenuation and sound source directivity.

In Version 1 of MPEG-4, there is no normative support for the structure of upstream data or its semantics. Version 2 standardizes both the mechanisms with which the transmission of such data is triggered at the terminal, as well as its formats as it is transmitted back to the sender. The inclusion of a normative specification for backchannel information in Version 2 closes the loop between the terminal and its server, vastly expanding the types of applications that can be implemented on an MPEG-4 infrastructure.

The BIFS Scene description framework, described in the previous section, offers a parametric methodology for scene structure representation in addition to efficiently coding it for transmission over the wire. Version 2 of the MPEG-4 standard also offers a programmatic environment, in addition to this parametric capability. Version 2 defines a set of Java™ language APIs (MPEG-J) through which access to an underlying MPEG-4 engine can be provided to Java applets (called MPEG-lets). This tool forms the basis for very sophisticated applications, opening up completely new ways for audio-visual content creators to augment the use of their content.

Finally, MPEG-4 Systems completes the toolbox for transport and storage of MPEG-4 content by providing:

- A file format [9] for the exchange of MPEG-4 content. This file format provides meta-information about the content, in order to allow indexing, fast search and random access. Furthermore it is possible to easily re-purpose the file content for delivery over various delivery layers.
- The specification of MPEG-4 delivery on IP, jointly developed with IETF. This Internet-Draft will specify the encapsulation of MPEG-4 data in RTP, the Real-time transport protocol, which is the focal point for delivery of streaming multimedia content on the Internet.
- The specification of MPEG-4 delivery on MPEG-2 Systems [2]. This amendment for MPEG-2 Systems will define a set of descriptors that provide for the signaling of the presence in the MPEG-2 multiplex of MPEG-4 content, both in the form of individual MPEG-4 elementary streams as well as complete MPEG-4 presentations. MPEG-2 content and MPEG-4 content may not only co-exist within an MPEG-2 multiplex, but it is also possible to reference the MPEG-2 content in the MPEG-4 scene description.

5.2. MPEG-4 Systems and competing technologies

This section aims to complete the description of MPEG-4 Systems by trying to make a fair comparison between the tools provided by MPEG-4 Systems and the ones that can be found – or will be found in a near future – in applications in the market place.

Technical issues aside, the mere fact of being proprietary is a significant disadvantage in the content industry when open standard alternatives exist. With the separation of content production, delivery, and consumption stages in the multimedia pipeline, the MPEG-4 standard will enable different companies to separately develop authoring tools, servers, or players, thus opening up the market to independent product offerings. This competition is then very likely to allow a fast proliferation of content and tools that will inter-operate.

5.2.1. *Transport*

As stated in Section 4.4, MPEG-4 Systems does not specify or standardize a transport protocol. In fact, it is designed to be transport-agnostic. However, in order to be able to utilize the existing transport infrastructures (e.g. MPEG-2 transport or IP networks), MPEG-4 defines an abstraction of the delivery layer with specific mappings of MPEG-4 content on existing transport mechanisms [7]. However, there are two exceptions in terms of the abstraction of the delivery layer: the *MPEG-4 File Format* and the *FlexMux* tool.

There are several available file formats for storing, streaming, and authoring multimedia content. Among the ones most used presently are Microsoft's ASF (advanced streaming format), Apple's QuickTime, as well as RealNetworks' file format (RMFF). The ASF and QuickTime formats were proposed to MPEG-4 in response to a call for proposals on file format technology. QuickTime has been selected as the starting point for the collaborative development of the MPEG-4 file format (referred to as 'MP4') [9]. The RMFF format has several similarities with QuickTime (in terms of object tagging using four-character strings and the way indexing information is provided). MP4 inherits from QuickTime key technical features such as the ability to stream content from multiple sources (local or through a network) with interactivity and known rendering quality. In addition to these key features, the MP4 format adds support for MPEG-4 specific features. In particular, MP4 inherits from MPEG-4 all the new and compelling audio, video and systems multimedia content.

The MPEG-4 proposal for light-weight multiplexing (FlexMux) addresses some MPEG-4 specific needs as described in Section 4.4. The same kinds of requirements have recently been raised within the IETF with regards to delivery of multimedia content over IP networks. With the increasing number of streams resident in a single multimedia program, with possibly low network bandwidths and unpredictable temporal network behavior, the overhead incurred by the use of RTP streams and their management in the receiving terminals is becoming considerable. IETF is therefore currently investigating a generic multiplexing solution, that has requirements similar to that of the MPEG-4 FlexMux. We expect that, with the close collaboration between IETF and MPEG-4, a consistent solution will be developed for the transport of MPEG-4 content over IP networks.

5.2.2. *Streaming framework*

With the specifications of the object description framework and the sync layer, MPEG-4 Systems provides a consistent and efficient framework for the description of content and the means for its synchronized presentation at client terminals. At this juncture, this framework, with its flexibility and dynamics, does not have any equivalents in the standards arena. A parallel could be drawn with the combination of RTP and SDP (session description protocol); however, such a solution is Internet-specific and cannot be applied directly on other systems like digital cable or DVDs (digital video disc).

5.2.3. *Scene description representation*

Within the context of Web applications, a number of competitors to MPEG-4 BIFS base their syntax architecture on the XML (Extensible Markup Language) [1] syntax, while MPEG-4 bases its syntax architecture on VRML using a binary, SDL-described form (MPEG-4 Syntactic Description Language[3]). The main difference

[3] SDL originates from the need to formalize bit-stream representation. SDL is an intuitive, natural and well-defined extension of the typing system of C++ and Java. It has been explicitly designed to follow (as much as possible) a declarative approach to bit-stream syntax specification. SDL-based specifications describe how the data is laid out on the bit-stream. SDL makes the task of understanding the specification and transforming it into running code straightforward, like it is done in other domains (e.g., Sun ONC RPC or OMG CORBA).

between the two is that XML is a general purpose text-based description language for tagged data, while VRML with SDL provide a binary format for a scene description language.

The competition is first at the level of the semantics, i.e., the functionality provided by the representation. At the time the MPEG-4 standard was published, several specifications were providing semantics with an XML-compliant syntax to solve multimedia representation in specific domains. For example, the W3C (World Wide Web Consortium) HTML-NG (HTML next generation) was redesigning HTML to be XML compliant [12]. The W3C SMIL (Synchronized Multimedia Integration Language) working group has produced a specification for 2D-multimedia scene description [11]. The ATSC/DASE (advanced television systems committee/digital-TV application software environments) BHTML (broadcast HTML) specifications were working at providing broadcast extensions to HTML-NG. The Web3D Consortium X3D (extensible 3D) requirements were investigating the use of XML for 3D scene description, whereas W3C SVG (scalable vector graphics) was standardizing scalable vector graphics also in an XML compliant way [10].

MPEG-4 is built on a true 3D scene description, including the event model, as provided by VRML. None of the XML-based specifications currently available reaches the sophistication of MPEG-4 in terms of composition capabilities and interactivity features. Furthermore, incorporation of the temporal component in terms of streamed media, including scene descriptions, is a non-trivial matter. MPEG-4 has successfully addressed this issue, as well as the overall timing and synchronization issues, whereas alternative approaches are lacking in this respect.

A second level of competition can be seen in the coded representation of the scene structure (text-based versus binary representation). An advantage of a text-based approach is ease of authoring; documents can be easily generated using a text editor. Such textual representations for MPEG-4 content, based on extensions of VRML or on XML, were under consideration at the time this paper was published. However, for delivery and streaming over a finite bandwidth medium, a compressed representation of multimedia information is without a doubt the best approach from the bandwidth efficiency point of view. This is the problem primarily addressed and solved by MPEG-4. None of the XML-based approaches have addressed satisfactorily this issue up to now.

Indeed, in addition to XML semantics that leverage the functionality developed by MPEG-4, a complete competitive solution would also need to define a binary mapping as well as media streaming and synchronization mechanisms. In June 1999, there was no evidence how this could happen on a short or medium term schedule. Indeed, all the potential alternative frameworks are at the stage of research and specification development, while MPEG-4 is at the stage of standard verification and deployment. There is no evidence that any of these frameworks will be able to leverage efficiently all of the advantages of MPEG-4 specifics, including compression, streaming and synchronization. Finally, the fragmented nature of the development of XML-based specifications by different bodies and industries certainly hinders integrated solutions. This may therefore cause a distorted vision of the integrated, targeted system as well as duplication of functionality.

5.3. The key features of MPEG-4 systems

In summary, the key features of MPEG-4 Systems can be stated as follows:

1. MPEG-4 System first provides a consistent and complete architecture for the coded representation of the desired combination of streamed elementary audio-visual information. This framework covers a broad range of applications, functionality and bit-rates. However, not all of the features need to be implemented in a single device. Through profile and level definitions, MPEG-4 System establishes a framework that allows consistent progression from simple applications (e.g., an audio broadcast application with graphics) to more complex ones (e.g., a virtual reality home theater).
2. MPEG-4 System augments this architecture with a set of tools for the representation of the multimedia content, viz., a framework for object

description (the OD framework), a binary language for the representation of multimedia interactive 2D and 3D scene description (BIFS), a framework for monitoring and synchronizing elementary data stream (the SDM and the SyncLayer), and programmable extensions to access and monitor MPEG-4 content (MPEG-J).

3. Finally, MPEG-4 Systems completes this set of tools by defining an efficient mapping of the MPEG-4 content on existing delivery infrastructures. This mapping is supported by the following additional tools: an efficient and simple multiplexing tool to optimize the carriage of MPEG-4 data (FlexMux), extensions allowing the carriage of MPEG-4 content on MPEG-2 and IP systems, and a flexible file format for authoring, streaming and exchanging MPEG-4 data.

The MPEG-4 System tools can be used separately in some applications. But MPEG-4 Systems also guarantees that they will work together in an integrated way, as well as with the other tools specified within the MPEG-4 standards.

Acknowledgements

The MPEG-4 System specification reflects the results of teamwork within a worldwide project in which many people invested enormous time and energy. The authors would like to thank them all, and hope that the experience and results achieved at least matched the level of their expectations.

Among the numerous contributors to the MPEG-4 Systems projects, three individuals stand out for their critical contributions in the development of MPEG-4: Cliff Reader is the person that originally articulated the vision of MPEG-4 in an eloquent way and led it through its early formative stages; Phil Chou was the first to propose a consistent and complete architecture that could support such a vision; and Zvi Lifshitz who, by leading the IM1 software implementation project, was the one that made the vision manifest itself in the form of a real-time player.

Finally, the authors would like to thank Ananda Allys (France Telecom – CNET), for providing some of the pictures used in this paper; the European project MoMuSys for its support of Olivier Avaro and Liam Ward; the National Science Foundation and the industrial sponsors of Columbia's ADVENT Project for their support of Alexandros Eleftheriadis; the German project MINT for its support of Carsten Herpel; the General Instrument Corporation for their support to Ganesh Rajan.

References

[1] T. Bray, J. Paoli, C.M. Sperberg-McQueen (Eds.), Extensible markup language (XML) 1.0, 10 February 1998. http://www.w3.org/TR/REC-xml.

[2] ISO/IEC 13818-1, Generic Coding of Moving Pictures and Associated Audio Information – Part 1: Systems.

[3] ISO/IEC 14496-1, Coding of audio-visual objects: systems, final draft international standard, ISO/IEC JTC1/SC29/WG11 N2501, October 1998.

[4] ISO/IEC 14772-1, The Virtual Reality Modeling Language, 1997, http://www.vrml.org/Specifications/VRML97.

[5] ISO/IEC 14496-2, Coding of audio-visual objects: visual, final draft international standard, ISO/IEC JTC1/SC29/WG11 N2502, October 1998.

[6] ISO/IEC 14496-3, Coding of audio-visual objects: audio, final draft international standard, ISO/IEC JTC1/SC29/WG11 N2503, October 1998.

[7] ISO/IEC 14496-6, Coding of audio-visual objects: delivery multimedia integration framework, final draft international standard, ISO/IEC JTC1/SC29/WG11 N2506, October 1998.

[8] ISO/IEC JTC1/SC29/WG11 N2723, MPEG-4 requirements document, March 1999.

[9] ISO/IEC JTC1/SC29/WG11 N22739, Text of ISO/IEC 14496-1/PDAM1, March 1999.

[10] Scalable vector graphics (SVG) Specification. W3C working draft 11-February-1999. http://www.w3.org/TR/WD-SVG/.

[11] Synchronized multimedia integration language. (SMIL) 1.0 Specification W3C Recommendation 15-June-1998. http://www.w3.org/TR/REC-smil/.

[12] XHTML™ 1.0: The extensible hypertext markup language. A reformulation of HTML 4.0 in XML 1.0. W3C working Draft 24th February 1999. http://www.w3.org/TR/WD-html-in-xml/.

Olivier Avaro was born in Provence, France on 27 July 1968. He received his Dipl. Ing. degree in telecommunication engineering in 1992 from the Higher School of Telecommunication of Brittany. He joined France Telecom-CNET department on image communication in Paris in 1992. He worked on image representation techniques and analysis. His research areas cover video compression algorithms, error resiliency of video compression algorithms, invariant representation of images and shapes and model based representation for interpersonal communications. He has been early involved in the ISO/MPEG-4 project, in particular through the European platform MAVT and its successor MoMuSys. He later managed the France Telecom CNET project Oxygen on MPEG-4 multimedia teleconferencing systems. In 1999, he joined Deutsche Telekom – Berkom department on Multimedia Communication in Darmstadt within the context of joint Deutsche Telekom – France Telecom projects. He is currently chairman of the MPEG-4 Systems sub-group.

Alexandros Eleftheriadis was born in Athens, Greece, in 1967, and received his Ph.D. in Electrical Engineering from Columbia University in 1995. Since 1995 he has been an Assistant Professor at Columbia, where he is leading a research team working on media representation with emphasis on multimedia software, video signal processing and compression, video communication systems and the mathematical fundamentals of compression. Dr. Eleftheriadis is a member of the ANSI NCITS L3.1 Committee and the ISO-IEC JTC1/SC29/WG11 (MPEG) Group, where he serves as the Editor of the MPEG-4 Systems specification. His awards include an NSF CAREER Award.

Carsten Herpel was born in Cologne, Germany in 1962. He received his diploma in communication engineering from RWTH Aachen, Germany, in 1988. He then joined Thomson Multimedia's research facility in Hannover, Germany, and developed video coding algorithms, with a focus on scalable coding tools. His research interests have then moved to multimedia systems with a current focus on MPEG-4. Mr. Herpel serves as a co-editor to the MPEG-4 Systems specification.

Ganesh Rajan received his Ph.D. degree in Electrical Engineering from the University of California, Santa Barbara, in 1987. He was with the Tektronix Laboratories, Tektronix, Inc., until October 1995. During his tenure there, he led the Digital Signal Processing group in projects encompassing the areas of digital signal and image processing, and digital communications. From October 1995 until May 1999, he was with the Advanced Technology group, General Instrument Corp. and was involved in the areas of multimedia coding and presentation architectures. Ganesh has been involved with the MPEG related activities since early 1996, especially in the Synthetic and Natural Hybrid Coding (SNHC), and the Systems subgroups. He was the chairman of the SNHC group during the 36th ISO/IEC JTC1/SC29/WG11 (MPEG) meeting at Chicago, in October 1996. He is the co-editor of the ISO/IEC 14496-1 (MPEG-4 Systems) specifications. Ganesh is a member of the IEEE and the ACM.

Liam Ward graduated from Dublin City University (Ireland) in 1991 with a degree in Electronic Engineering. He is currently a Senior Research Officer with R&D consultants Teltec Ireland. He is an editor of the recently completed MPEG-4 Systems specification, in particular the part dealing with scene description (BIFS). He is also leader of the European ACTS-DICEMAN consortium, a project which is developing techniques for the trading of audio-visual material on digital networks using MPEG-7.