Towards a uniform management of multimedia presentations with MHEG

H. Müller, B. Jandl
Joanneum Research, A-8010 Graz, <mue|jandl>@iis.joanneum.ac.at

W. Herzner, M. Kummer
Austrian Research Centre Seibersdorf, A-2444 Seibersdorf,
<herzner|kummer>@zdfzs.arcs.ac.at, http://www.arcs.ac.at

Abstract. While for monomedia content categories like text, graphic, images, audio and video, numerous encoding formats exist for interchange of data between applications, there are almost none such formats for whole multimedia presentations including layout, interaction, contextual and behavioral aspects. The international standard MHEG is one of the few attempts in that direction. After outlining the basic features of MHEG, this papers presents the results of the ESPRIT project OMHEGA, which developed a distributed architecture as well as the related services for the lifecycle management of multimedia objects (i.e. creation and modification, storage, retrieval, distribution and presentation) based on MHEG, and closes with a perspective into the future.

Keywords. IDL, hypermedia, MHEG, multimedia, multimedia objects, OMHEGA, presentation behaviour.

1 Introduction

Today, multimedia information and applications can be regarded as integral part of computer environments. From the beginning of this development there was a request for common formats to enable the exchange of information between different applications and platforms, which has strongly been with the availability of global computer networks. For monomedia data, a number of such formats exist today, either as international or as industrial standards.

However, for complete presentations (including contents, layout and behaviour), the situation is different: a pretty large number of authoring tools are available, some of them even fairly platform independent, but it is rather impossible to exchange information ('applications') between those tools, or between authoring and presentation tools (i.e. 'browsers'). Often, applications are finally output as executable code, which is then again be platform-bound.

Hence, a standardized encoding for such multimedia information is needed, which is not only suited for the transmission of data from authoring tools to browsers, but also between authoring tools. MHEG [ISO96] has been developed for just that purpose.

In the next chapter, MHEG and its basic features are introduced. In chapter 3, the ESPRIT project OMHEGA is presented, which developed a model for lifecycle management of so-called MHEG objects, where emphasis is laid on the specified services and the methods used for the 'proof of concepts'. The final chapter addresses future developments like the usage of MHEG in settop boxes and the integration with WWW.

2 MHEG

'MHEG' stands for "multimedia/hypermedia expert group". It is developed within the same common subcommittee of ISO (*International Standardization Organization*) and IEC (*International Electrotechnical Committee*), which also developed JBIG [ISO93a], JPEG [ISO94a], MPEG [ISO93b], and MPEG-2 [ISO95]. Experts from states like France, Germany, Italy, Japan, UK, and USA participated (and are still participating) in this standardization activity, including Austria, where for instance, several of the authors contributed since about 1992.

MHEG has been designed as multipart standard. Part 1 specifies general concepts and an encoding in ASN.1 ([ISO94b], [Rose90], an International Standard for binary encoding of structured information), while further parts address either extensions (e.g. part 3 for scripting), profiles (e.g. part 5 for settop unit environments), or further components like APIs. A part *n* is usually referred to as 'MHEG-n'.

MHEG carries the ISO-number 13522, but is also designated as the recommendation series T.170 from ITU-T (*International Telecommunication Union, Telematic sector*). So, ISO/IEC 13522-1 (MHEG-1) is also T.171, MHEG-5 also T.172, and MHEG-3 also T.171.

In the following, the basic concepts and features as specified in MHEG-1 shall only shortly be summarized, since they have already been described in earlier papers, e.g. [CoBe94] or [Pric93] (although MHEG-1 has been modified in details since then) . At the end of this chapter, these will be compared with related standards and systems. MHEG-3 and MHEG-5 will also quite shortly be addressed.

2.1 Concepts and Features

As already mentioned, MHEG serves for the specification of multimedia representations, where it is not sufficient to specify *what* shall be presented (content) and *where* (layout), but also *when* and *how* (behaviour), including interaction with the user.

Object-Orientation

In order to accomplish this in an easy to handle though flexible way, MHEG groups data for multimedia presentations into so-called 'objects': well-defined pieces of information, each of them serving for a specific purpose. MHEG takes an object-oriented approach with respect to inheritance and polymorphism; that means that classes inherit attributes and behaviour from their super-classes, and operations (called 'actions' in MHEG) applicable to some class are also applicable to all its sub-classes. As illustrated in fig. 1, MHEG-1 defines 11 (basic) object classes.

As usual, abstract classes cannot be instantiated. The root class, *MH-Object*, defines elements common to all classes: attributes for object identification and description (e.g. creation date or copyright), and the most basic behaviour like preparation and destruction.

Descriptors provide additional information about other objects, mainly human readable, and *containers* allow to group MHEG objects loosely (e.g. to indicate which objects are needed together for an application), but without further predefined semantics. The remaining classes are shortly introduced below.

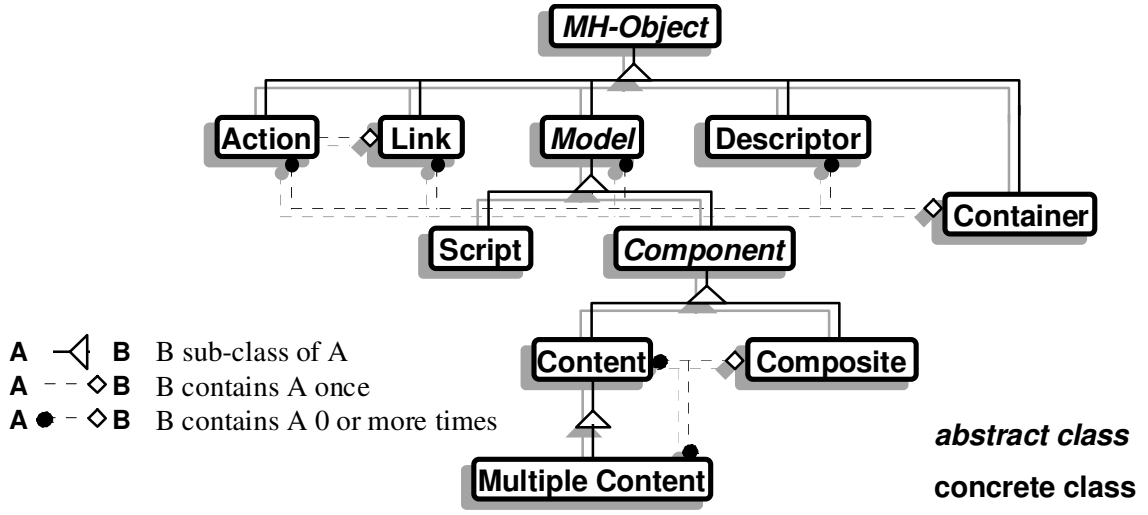

Fig. 1. MHEG-1 object class hierarchy

Behaviour.

MHEG's *link objects* specify under which conditions *action objects* are to be executed. Conditions are boolean combinations of 'trigger conditions' over events, and 'constraint conditions' over object states and attributes. For instance, a condition could say "when the speed of object M1 was < 1 and becomes ≥ 2 and (M2 is visible or M3 is running)". Whenever a *link's* condition becomes satisfied, its 'effect', namely an *action object*, becomes executed.

Action objects describe what has to be done when they are executed. They consist of 'elementary actions' or nested action objects, and a flag indicating whether its component actions shall be executed in sequence or parallel. For instance, the (elementary) action (e.a. for short) "set speed ((M1, M2), 150%, 2000)" increases the display speed of the objects M1 and M2 by 50% within 2000 milliseconds.

Finally, *script objects* allow to specify what cannot be expressed directly within MHEG (e.g. computations). We will come back to scripts when discussing MHEG-3.

Contents.

Subclasses of the abstract *component class* serve for the specification of content. While *content objects* represent elementary contents like a JPEG image or a 'WAV' sound, and *multiple content objects* represent multiplexed content like MPEG or 'AVI' streams, *composite objects* allow to group and position components relative to each other. (They include their components by reference.) See also the clause about space and time below.

Similar to composites, content objects will usually refer to the physical data rather than contain them directly.

Space and time.

Each component object has its own 4-dimensional co-ordinate space associated: three spatial axes, and a time axis (with milliseconds as units). Initially, static objects have infinite duration, and composites have the minimum extent covering all their components in space and time.

For each component of a composite, an arbitrary axis-parallel subspace is projected into the parent's space, which is controlled by a number of appropriate e.a. (Since composites can again contain composites, this applies recursively.)

Along a time axis, the 'current temporal position' (CTP) indicates that part of the associated component object (e.g. frame) which is currently presented. In addition, so-called 'timestones' can be placed along the time axis. Whenever CTP passes a timestone, an appropriate event is raised, which can be recognized in link conditions. Hence, timestones can be used for synchronization.

Interaction and anchors.

Conceptually, the user can interact with each presented object in two modes, by selection or by modification. How the latter is performed, depends on its type and content, and is handled by the user interface. However, selection and modification completion can be recognized in link conditions.

To allow for selection of parts of content objects, e.g. some substring of a text, or a closed area of an image, MHEG-1 provides the concept of 'anchors'. These are content objects which specify selectable elements of other component objects rather than presentable content. Once assigned to a component object, the identified part of that object will be indicated as anchor to the user. Its selection will make the anchor object selected (which again can be recognized in link conditions).

Channels.

A channel is a presentation means on any node of a computer network, for example a window and/or the loudspeaker on a user's PC. An arbitrary number of channels may be open at any time, allowing for distributed, multiple user presentations (and applications).

Part 3: scripting

The already mentioned script objects essentially contain a script, i.e. pieces of code in some (script) language. MHEG allows to use foreign languages like OpenScript [Asym94] or Java [GoMc95], or even programming languages like C, but leaves their interpretation up to the presentation environment.

As common base for all MHEG-1 implementations, MHEG-3 specifies a p-code like, stack-oriented language, called 'SIR' (*script interchange representation*), together with the corresponding virtual machine for semantically definition.

Part 5: profile for STU-applications

The consortium DAVIC (Digital AudioVIsual Council) [DAVIC95], an association of the about 200 largest players in the field of consumer electronics, which is currently developing a complete system architecture for settop unit environments, has decided to use a subset of MHEG-1 for their application level protocol. This subset, together with a number of additional sub-classes of MHEG objects, is specified in MHEG-5.

2.2 Related Standards

HTML and Java

In the WWW, information is scattered over information objects (i.e. files) like HTML-pages or GIF-images in WWW with many hyper-links between them. Similarily, information is distributed over MHEG-objects with a lot of references between them. But while MHEG allows for detailed specification of presentation behaviour and layouting by the author, HTML [BeCo95] gives much more freedom to the browsers referring to this; and the presentation of individual information units (e.g. HTML-pages and MPEG-videos) is almost context-free or rigidly related (e.g. embedded images are either shown on display of their page or not), leaving not much freedom for the author to control presentation behaviour. Even HTML 3.0 will not change too much in that direction. Java-applets allow for detailed behaviour control, though, but again act widely context-free relative to other applets or the whole presentation. To some degree, they can be compared with MHEG's script objects.

In HTML, interaction occurs either through selection of anchors, or through so-called forms. Anchor selection is handled in the browser, while form input is sent back to server, where it is interpreted. In MHEG, all user input is handled by the so-called MHEG engine (some sort of browser, see chapter 3), but the user interface can look quite similar to that of WWW, especially for MHEG-5.

HyTime

HyTime [Gold91a], [ISO92b] is an ISO-standard specifying 'architectural forms' for SGML DTDs (like HTML) [Brya88], [Gold91b], [ISO86], [ISO88], which enable the specification of spatio-temporal relationships among information objects in a complete form with respect to synchronization (i.e. end-synchronization can be expressed by simple 'right justification' along a time-axis. Therefore, HyTime-encoded objects request scheduling before presentation (which is not necessary for MHEG, because it is strictly event-driven). On the other side, HyTime provides little support for expressing interaction, because it is not event-driven.

Interpretation of HyTime-documents is performed be HyTime-engines, as MHEG-documents are presented by MHEG-engines. HyOctane [BuEA 94] is a realisation of such an HyTime-engine, which provides an programming interface for application control (including interaction).

It could be said that from a conceptual view HyTime is located 'above' MHEG, because the output of a HyTime scheduler could be expressed using MHEG.

Authoring systems

Authoring systems like Macromedia director [MaDi95] or Multimedia ToolBook [Asym94] today usually provide most of the features of MHEG. However, applications created with such tools are mostly produced as monolithic executables, hence platform dependent and large, and sometimes need at least a run-time version of the system to be presented, which hampers distribution.

3 OMHEGA

For the life-cycle management of MHEG-objects (creation, storage, retrieval, presentation) it is helpful to define a set of services, which should be based on common architectural models. In the OMHEGA project (Open MHEG Architecture) a system architecture, which can serve as a base of reference for future multimedia and hypermedia application developments in open distributed environments, was developed. Out of a requirement analysis and the definition of six basic application areas a generic reference model was defined. Based on this reference model, a list of thirty services was specified by functional descriptions and formal specifications of the APIs. This chapter gives an overview about the OMHEGA services, their utilisation in a MHEG engine, and methods to visualise and proof the interaction between different service groups. The specification of a specific API will be shown in an example.

The services that are needed by the user are typically expressed very differently depending on the user experience and environment. In the OMHEGA project all specifications and developments are based on the approach of 'basic applications'. Across all sectors we can identify six applications categories: Database Retrieval, Telemeeting, Collaborative Editing, Messaging, Distribution and Teleaction. In this model specific applications can be constructed by using one or a combination of several basic applications. For example, a telemedicine application my be composed of a telemeeting and collobarative editing part.

Database Retrieval involves the sending of a query for a specific item of information and the return of a result plus all the necessary control information. The information items used within the OMHEGA services are either MHEG objects or content objects (text, audio, video, etc.) that are referenced by MHEG objects.

Telemeeting applications involves remote participants accomplished via real time telecommunication means. OMHEGA services can be used in telemeeting applications to specify the necessary user interfaces and to control the presentation of live audio and video data.

Colloborative Editing involves the working together of two or more users in order to perform a single task. OMHEGA services allow for a consistent presentation of MHEG objects in jointly manipulated documents at all participating user sites.

Messaging involves the storage, distribution and retrieval of information generally separated by a time delay. OMHEGA services can be used for consistent presentation of the jointly manipulated documents at all participating user sites.

Distribution covers information delivery without interaction, including broadcast and multicast. OMHEGA services allow for dissemination from one sender to one or several end-users, with an aspect of quasi synchronisation for deliveries.

Teleaction involves another process after remote observation, measurement or activation. It includes teleobservation, telemeasure, teleordering, teleprinting, etc. OMHEGA services can be used in a similar way as for telemeeting.

According to the life cycle of an MHEG object, services have been grouped into three categories, see Fig.2: Composition oriented services, Distribution oriented services and Presentation oriented services. Beside these categories a set of Utility services facilitates the operation of various tasks.

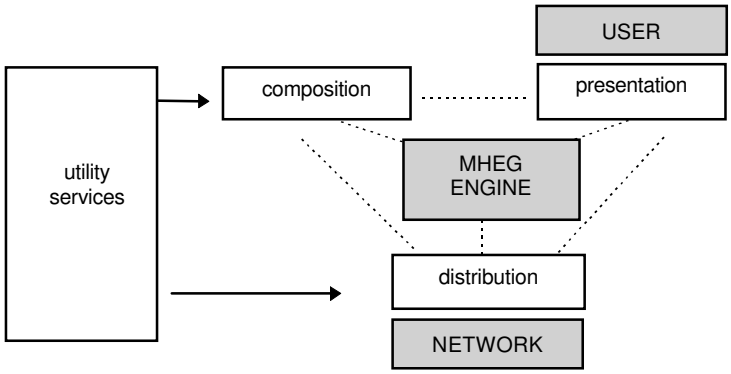

Fig. 2. Overview on service categories

The MHEG engine is the driving force in a MHEG runtime system. It receives MHEG objects, decodes them and performs certain behaviour associated with the objects such as link and action execution, synchronisation, content display and user interaction management. The MHEG engine keeps track of the MHEG objects and presentation status, and the events occurring during the presentation. The MHEG engine description does not belong to the normative part of the standard itself and consists of an abstract entity, which is used to specify a MHEG runtime environment.

Within OMHEGA, some services are closely related to the functionality of a MHEG engine. These are in particular the Interpretation Service, the Object Handling Service and the Starter Service. Other utility services like the MHEG CoDec Service and the Grammar Support Service are also directly used by the MHEG engine. There is no MHEG engine service defined within OMHEGA since some of the services are not used for the MHEG engine operation. Services, needed to build an MHEG engine as assumed by the MHEG standard are:

Distribution Services - The set of distribution oriented services provides the distribution of MHEG objects and content data in a distributed environment keeping storage, encoding, location, security and accounting matters transparent from the using client. The Storage Services Group facilitates storing and administration of different types of data on a persistent storage device. This data is transferred over a network by the Transport Services. The Communication Services are on a higher level and therefore hide location, security and accounting matters.

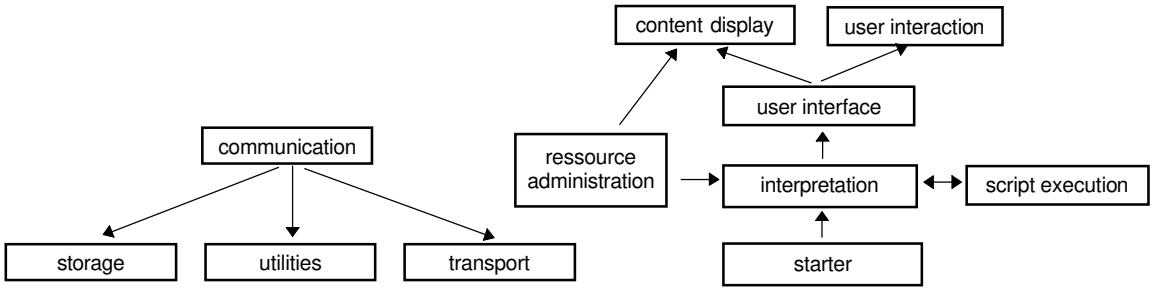

Figure 3: Overview on Distribution and Presentation services

Presentation Services - These services take over the presentation of multimedia data to the user. The Presentation Services Group is responsible for the interpretation of various types of data. Thus it triggers the presentation process, which is supported by the User Interface Service. MHEG engines could make extensive use of these services.

Composition Services - The set of composition oriented services facilitates the creation

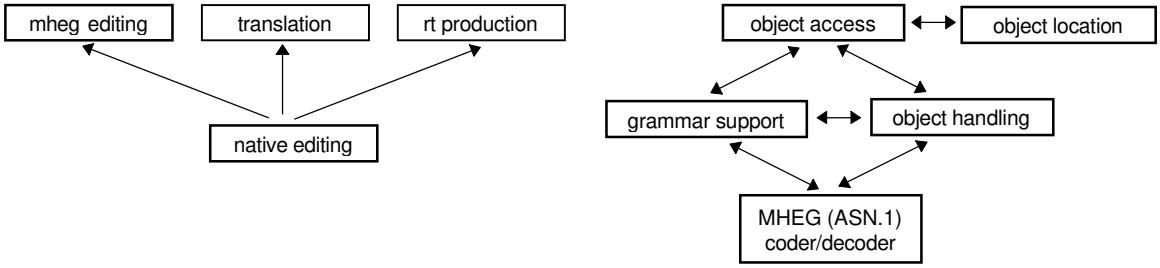

Figure 4: Overview on Composition and Utility services

and altering of MHEG objects and content data such as text, images and video. The MHEG Editing Service supports the composition of MHEG objects through direct interaction with the user. The Translation Service enables the creation of MHEG objects out of various other formats while the Real-time Production Service facilitates the creation and altering of MHEG objects during run-time. These services are supported by the MHEG Native Editing Service.

Utility services - Their objective is to support the operation of the services which have been figured out before. Three groups of services have been identified: User Services Group, MHEG Object Management Services, and Encoding/Decoding Services.

The services are defined by their abstract interface, functional description and a language independent interface definition using the Interface Definition Language IDL defined by the Object Management Group) [IDL92], see Fig. 5.

Send MHEG Action

Pass a set of Interchanged Action Objects or a set of references to Interchanged Action Objects to a session and execute those actions on the targeted (and referenced) MHEG Interchanged Objects or Run-time Objects. The parameters are:

in:

- List of MHEG-Interchanged-Action-Object,
- List of Mh-Object-Reference (alternative)
- List of References on Mh-Object-References (optional)
- Session-id.

out: none

Exceptions raised:

wrong_state,
invalid_object,
invalid_reference,
invalid_session_id,
insufficient_ressources

```
module OMHEGA_PRE
{
#include "glb_type.idl"
.......

interface INT // MHEG Interpretation
Service
{  void start_interpretation
     (in  RESOURCE_LIST ResourceList,
      out SESSION_ID    SessionId)
   raises
      (exWrongState,
       exInvalidSessionId,
       exInsufficientResources);
   .......
   .......
   void send_mheg_action
     (in L_MHEG_ACTION ActionList,
      in L_OBJECT_REF ObjRefList,
      in SESSION_ID SessionId)
   raises
     (exWrongState,
      exInvalidObject,
      exInvalidReference,
      exInvalidSessionId,
      exInsufficientResources);
      .......
      .......
};
// end of definition:
// MHEG Interpretation Service
```

Abstract definition **IDL specification**

Figure 5: Abstract definition and IDL specification of a service element

'Proof of concepts' and validation of the services was done by the modelling of basic telematic applications (database retrieval, telemeeting, collaborative editing, messaging, distribution, teleaction). This was done by the help of diagrams, using a graphical notion similar to the Booch method [Booc94], see Fig. 6, and by an interactive visualisation. The so-called Service Visualizer interprets a script and displays the simulated application in a graphical way as shown in Fig. 7. In a logger window all the individual operation together with the calling parameters are listed.

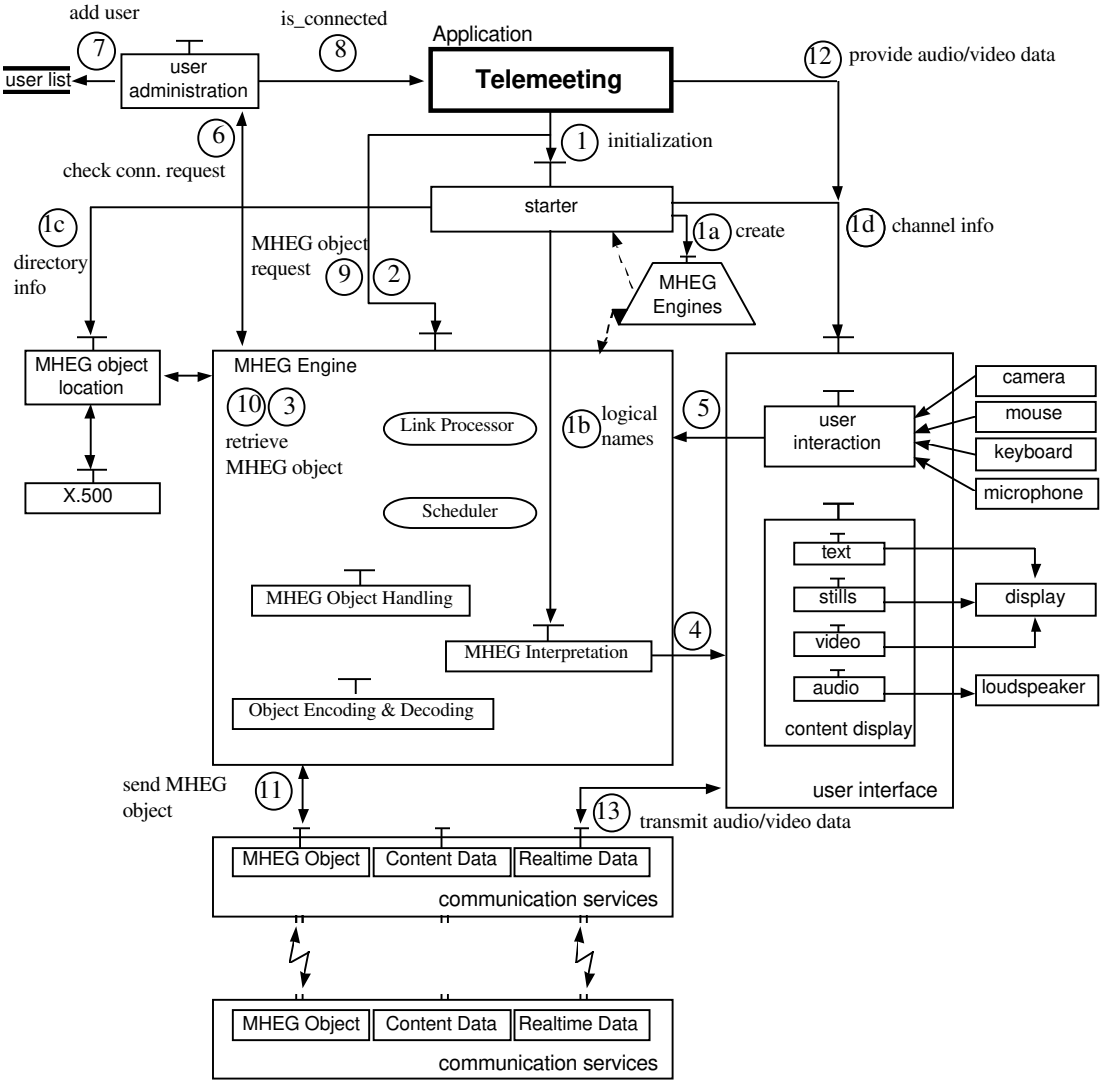

Fig. 6: Components and data flow of a telemeeting application

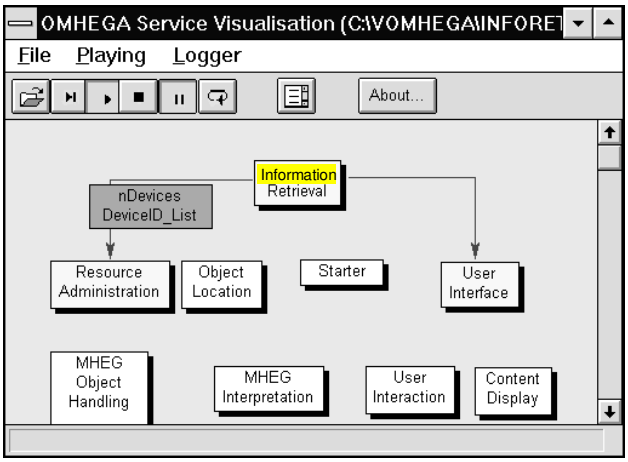

Fig. 7: Service visualizer

4 Future Developments

As already mentioned in chapter 2, MHEG-5 has been selected by DAVIC for the encoding of application presentation within the framework for settop box systems. A settop box is an extension unit for conventional TV or HDTV, which allows by means of a remote to consume television/telecommunication services like video-on-demand, new-on-demand, teleshopping, homebanking, or distance learning. A number of market research experts expect a market potential of several billion dollars in the last years of this millennium for this telematic sector.

While settop boxes will contain MHEG presentation services, perhaps in silicon, and the distribution services will be realized by the corresponding DAVIC components, there is a need for production services or, in terms of OMHEGA, *composition services*. (Due to larger number of aspects to be considered during authoring in comparison with HTML, and the binary ASN.1-encoding of MHEG objects, it is not feasible to edit MHEG-objects directly.) At least three methods for the production of MHEG-objects exist: dedicated editors, adaptation of existing authoring systems to read/write MHEG-objects, and gateways converting foreign multimedia data into MHEG objects, perhaps on the fly. For all three, prototypes exist or are under construction. It can be assumed that during this year authoring tools will become publicly available. About that time, single-user MHEG-engines, presumably with a number of functional restrictions, should become available as well on PC, and Unix platforms.

Another issue concerns the relationship between MHEG and WWW including Java. There are ongoing discussions between ISO and IETF (*Internet Engineering Task Force*) on that matter. It can be expected that at least referencing MHEG-objects from HTML-pages and vice versa will be possible. In addition, it has recently been decided to use Java as script language for MHEG-5 script objects (more precisely: the virtual machine (VM) of Java is used as base for the so-called MHEG-5 VM, with an appropriate adaption of the Java language libraries). Finally, at least one implementation of an MHEG(-5) engine in Java is under development, which would allow to interpret MHEG-objects from HTML-pages in a direct way, although downloading an MHEG-engine als Java applet will take considerable time on existing networks.

Only future will know how much MHEG will play its role in next generation multimedia applications. Considering its potential in expressing spatio-temporal relationships, MHEG can contribute significantly to the distribution of application areas like CBT computer-based multi-user games, this the more when OMHEGA-results are exploited.

References

[Asym94] Multimedia ToolBook 3.0 und OpenScript Referenz, Asymetrix GmbH, 81677 Munich, Germany, 1994.

[BeCo95] Berners-Lee, T., Connoly, D.: Hypertext Markup Language - 2.0, Internet-Draft ('work in progress'), MIT/W3C, Mass., USA 1995

[Booc94] Booch, G.: Object-oriented Analysis and design with Applications, Benjamin Cummings, Redwoord City, CA 1994

[Brya88] Bryan M., SGML - an author's guide to the Standard Generalized Markup Language, Addison-Wesley, 1988

[BuEA 94] Buford J.F., Rutledge L., Rutledge J.L., Toward Automatic Generation of HyTime Applications, Proc. of the Eurographics Symposium on Multimedia/Hypermedia in Open, Distributed Environments, Graz/Austria 1994, Springer, 1994, pp.57-76

[CoBe94] Colaitis F., Bertrand, F., The MHEG Standard: Principles and Examples of Applications, Proc. of the Eurographics Symposium on Multimedia/Hypermedia in Open, Distributed Environments, Graz/Austria 1994, Springer, 1994, pp.3-17

[DAVIC95] DAVIC 1.0 Specifications, DAVIC Secr., Via Servais 125, I-10146 Torino, Italy 1995

[Gold91a] Goldfarb, C.F., HyTime: A standard for structured hypermedia interchange, IEEE Computer, August 1991, pp. 81-84.

[Gold91b] Goldfarb, C.F., The SGML Handbook, Oxford University Press, 1991.

[GoMc95] Gosling, J., McGilton, H.: The Java™ Language Environment: A White Paper, Sun Microsystems, Inc., Calif., USA 1995, <http://java.sun.com/whitePaper/java-whitepaper-1.html>

[IDL92] The Common Object Request Broker: Architecture and Specification, OMG Document Number 91.12.1, Revision 1.1, . Object Management Group, 1992.

[ISO86] ISO/IEC 8879:1986, Information processing systems - Text and office systems - Standard Generalized Markup Language (SGML), ISO/IEC, Geneve, Switzerland, 1986

[ISO88] ISO/IEC 9069:1988, Information processing systems - SGML support facilities - SGML Document Interchange Format (SDIF) , ISO/IEC, Geneve, Switzerland, 1988

[ISO92a] ISO/IEC 8632:1992, Information processing systems - Computer graphics - Metafile for the storage and transfer of picture description information , ISO/IEC, Geneve, Switzerland, 1992

[ISO92b] ISO/IEC 10744:1992, Information Technology - Hypermedia/Time-based Structuring Language (HyTime), ISO/IEC, Geneve, Switzerland, 1992

[ISO93a] ISO/IEC 11544:1993, Information technology - Coded representation of picture and audio information - Progressive bi-level image compression , ISO/IEC, Geneve, Switzerland, 1993

[ISO93b] ISO/IEC 11172:1993, Information technology - Coding of moving pictures and associated audio for digital storage media up to about 1,5 Mbit/s, ISO/IEC, Geneve, Switzerland, 1993

[ISO94a] ISO/IEC 10918:1994, Information technology - Digital compression and coding of continuous-tone still images, ISO/IEC, Geneve, Switzerland, 1994

[ISO94b] ISO/IEC 8824:1994, Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), ISO/IEC 8825:1994, Information technology - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) ISO/IEC, Geneve, Switzerland, 1994

[ISO95] ISO/IEC 13818, Information technology - Generic coding of moving pictures and associated audio information, ISO/IEC, Geneve, Switzerland, 1995

[ISO96] ISI/IEC 13522, Information technology - Coding of multimedia and hypermedia information, ISO/IEC, Geneve, Switzerland, 1996

[MaDi95] Macromedia Director 4.0, Macromedia Inc., Berkshire, UK 1995

[MeEA91] Meghini C., Rabitti F., Thanos C., Conceptual Modeling of Multimedia Documents, IEEE Computer, October 1991, pp. 23-30.

[Mumf89] Mumford A.M., The CGM Today and Tomorrow, Computer Graphics Forum 8, 2, 1989, pp. 125-128

[Pric93] Price R., MHEG: An Introduction to the future International Standard for Hypermedia Object Interchange, Proc. of the ATM Multimedia 1993, Anaheim, CA.

[Rose90] Rose M.T., The open book: a practical perspective on OSI, Prentice Hall, 1990.