INTEROPERABLE ITV SYSTEMS BASED ON MHEG

H.Cossmann, C.Griwodz, G.Grassel, M.Pühlhöfer, M.Schreiber, R.Steinmetz, H.Wittig, L.Wolf

IBM European Networking Center, Vangerowstr.18, 69115 Heidelberg, Germany

ABSTRACT

Today's interactive television systems are using proprietary communication protocols and interchange formats. To provide interoperability at the application level the next generation of interactive television system will be based on standardized communication protocols, monomedia and multimedia formats. This paper presents the Globally Accessible Services (GLASS) system which is a prototype interactive television system based on the Multimedia and Hypermedia Expert Group (MHEG) standard. After a brief introduction to MHEG as the multimedia interchange format between application server and set-top box in interactive television systems, the GLASS clients and servers are described, and an example scenario for navigation in the GLASS system is provided.

Keywords: interactive television, multimedia systems, multimedia applications, standards, portable application

1. INTRODUCTION

Global interactive television (iTV) systems have to be interoperable systems. To achieve this interoperability, common data formats and protocols are to be developed. One major insight gained by the iTV field trials is that standardization only at the level of monomedia information (e.g., MPEG, JPEG) is not sufficient to guarantee application interoperability in global iTV systems. Monomedia standardization does not address the interchange of multimedia and hypermedia information — the *look and feel* of the multimedia application. Therefore, the *ISO/IEC Multimedia and Hypermedia information coding Expert Group (MHEG)* concentrates on the definition of an interoperable application format between the application server and set-top box (MHEG part 5).

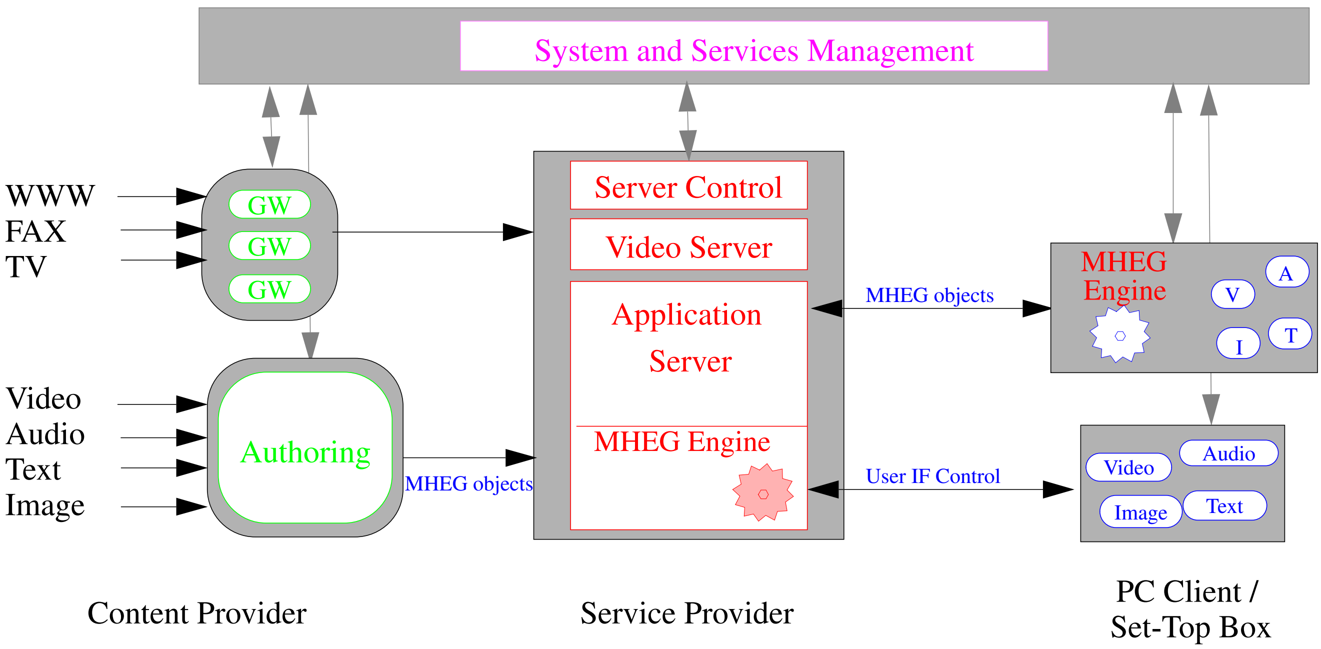

Fig. 1. : Architecture of the MHEG Based iTV System

In Figure 2 the architecture of an MHEG based iTV system is shown. In general, the following components of an interactive television system are affected by the MHEG standard:

(1) *content provider system:* Authoring components are used to produce applications in a format which complies to MHEG. There are converters for existing applications and authoring tools (e.g., Asymetrics Toolbook, Macromedia Authorware)

which have been used as MHEG front end authoring tool. Additionally, real-time converters provide application gateway functionalities (e.g., WWW<->MHEGgateway). Thereby, other interactive multimedia services are integrated.

(2) *service provider system:* MHEG engines are responsible for the parsing, interpretation and processing of MHEG applications. This includesthe output of the the monomedia contents and user interface event handling. An MHEG engine running at the application server implies a reduced workload for the small footprint client systems and an increased interaction/network load between the service provider and client system.

(3) *services management components*: Typical management services are implemented: session management (e.g., between MHEG clients and application servers, location management (e.g., MHEG object directory services), authentication and access control (e.g., control access to parts of an MHEG application), and billing.

(4) *clients:* The MHEG engine resides in the client system and is closely linked to the presentation subsystem. This allows fast interaction between the MHEG engine and presentation objects and reduces the network load.

(5) *distribution system:* If there is only one down-stream channel to the clients, multimedia and monomedia contents must be multiplexed in the delivery system. For example, MHEG applications are multiplexed with the MPEG-2 audio and video streams by using the specific identifier for MHEG data streams.

In the following sections, the basic GLASS system components are described: Section 2 introduces the general architecture, Section 3 describes the MHEG standard and its parts. Section 4 discusses the client and server components which build the MHEG run-time environment. Section 5 presents example applications. In Section 6, the DAVIC and GLASS architectures are compared. Section 7 concludes this article.

2. GLOBALLY ACCESSIBLE SERVICES SYSTEM

In the *Globally Accessible Services (GLASS)* project an interactive television system prototype based on MHEG-1 has been developed. This paper describes the GLASS system architecture and implementation. GLASS contains the following components of an interactive digital multimedia system: clients, application server, video server, system management, and gateways to other services (e.g., TV, radio, World-Wide-Web, electronic mail, FAX, BTX). Typical applications of the GLASS system are video on demand, television broadcast, multimedia presentation, games, tele-shopping.

The GLASS clients are running on DEC Alpha, Intel 80x86, Motorola 680x0, IBM POWER and Sun Sparc processors under the AIX, Linux, MacOS, DOS-Windows, OS/2, DEC UNIX and Solaris operating systems. The MHEG Engine is the central presentation processing component and resides in the client or in the application server. In the GLASS environment the clients have been used as the multimedia end-system.

3. OVERVIEW OF MHEG

MHEG is the name of the “Multimedia and Hypermedia information coding Expert Group”. This group is organized as working group 12 of the ISO/IEC Joint Technical Committee 1/ Sub-Committee 29. The standardization within MHEG is spit into several parts. Part 1 of the MHEG standard (MHEG-1) is the description of the “Coded Representation of Multimedia and Hypermedia Information Objects (MHEG)” in the Abstract Syntax Notation 1 (ASN.1). Due to the restricted resources of a set-top box, in part 5 of the MHEG standardization (MHEG-5) the subset of MHEG-1 objects for digital and interactive television applications has been defined. There are liaisons between MHEG and the Moving Pictures Expert Group (MPEG) to provide the transfer of MHEG objects in MPEG-2 streams, and between MHEG and the Digital Storage Media (DSM) Group to ensure that the DSM Command and Control (DSM-CC) standard provides communication between application servers and end-systems. MPEG and the DSM group are in the same sub-committee of ISO/IEC JTC1. The following section briefly introduces the role and overall concepts of the MHEG-1 and MHEG-5 standard.

3.1. General Concept

MHEG provides an interchange format for multimedia and hypermedia information and its machine-independent encoding for real-time multimedia applications, synchronization and interchange of applications. This includes temporal and spatial relationships between monomedia objects and user interaction mechanisms. MHEG is also a container and description format for each kind of monomedia data (e.g. bitmaps, text, video and audio).

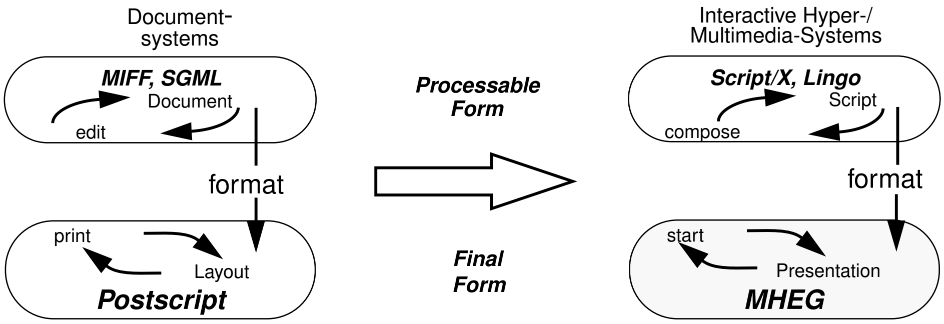

Fig. 2. MHEG as Multimedia/Hypermedia Presentation Format

In Figure 2 the analogy between conventional text processing systems and multimedia systems is shown. Comparable to Postscript as the standard page description language for linear documents, MHEG represents the common coding for multimedia and hypermedia applications. Similar to the transparency of Postscript for document authors MHEG is transparent for a multimedia designer. Multimedia and hypermedia applications are edited using existing multimedia authoring systems. For editing purposes these application are exchanged in a high-level scripting format (e.g., Lingo, Script/X). When the multimedia authoring process has been finished, the multimedia application is produced by converting the processable-form application to a final-form MHEG application. Thereby, content providers deliver portable MHEG applications once for a variety of client systems.

3.2. MHEG Structure

MHEG-1

MHEG-1 is the "Coded Representation of Multimedia and Hypermedia Objects (ASN.1)". It defines the general idea of MHEG: The representation is object-oriented. The MHEG standard defines classes of presentation objects. From the classes, MHEG objects may be instantiated by the presentation designer and interchanged between provider and end-system. This object representation form is called *interchanged objects*.

Part 1 of the standard defines also the semantics of the full MHEG class hierarchy and the ASN.1 encoding for the MHEG classes. The classes defined by MHEG-1 are created for distributed, interactive multimedia presentations and thus, the interaction facilities defined there are mainly event-driven. These events are, for example, generated by timers, user-input devices, embedded stream events, arriving objects, activation states of objects, or presentation system activity. This allows for powerful means to describe object interaction. It does not give the presentation author full programming functionality.

MHEG-2

MHEG-2 was supposed to provide an alternative SGML encoding of the MHEG objects defined by MHEG-1. For unknown reasons, this activity was pending for a long time and MHEG-2 has now been cancelled officially in the last ISO meeting.

MHEG-3

MHEG-3 defines scripting extensions for MHEG-1. The goal of MHEG-3 is the definition of extensions that provide full programming language capabilities in combination with MHEG-1. This means that a virtual machine model is defined that allows to use a stack machine model, local variable definitions, arithmetic and logical operation.

MHEG-4

MHEG-4 is the numbering authority of the MHEG standard. New monomedia standards, modifications to enumeration values, input event tables, explicitly defined content attributes like colour maps or fonts and additional class must be registered in MHEG-4.

MHEG-5

MHEG-5 is a sub-standard for low resources clients, and especially targetted at the set top unit area in interactive television systems. Although the initial idea was to define a subset of MHEG-1 to achieve this, activities driven by the Digital Audio-Visual Council (DAVIC) led to the definition of an independant class hierarchy.

When compared with MHEG-1, it takes some of MHEG-1's freedom of design from the author, restricts parallelity in a presentation and restricts object retrieval to specific container classes. On the other hand, MHEG-5 allows the usage of platform-dependent interface elements like scrollbars or button-styles, adds variables to the class hierarchy and adds a means to access system-specific interfaces.

MHEG-6

This part of standard is currently under consideration as a scripting extension which is adapted to the MHEG-5 standard.

4. ARCHITECTURE OF THE GLASS SYSTEM

As illustrated in Figure 2, giving an overview of the GLASS system architecture, the following components of an interactive television system are implemented in GLASS: end-systems, management components, application server, video server, and multimedia authoring systems. The GLASS system is a distributed system. Specific communication protocols allow for an distributed and heterogeneous system environment. Hence, each of the components can be located on a different computer system.

The GLASS end-system consists of the modules:

- MHEG Engine
- Control Agent
- User Interface Agent
- Presentation Objects

The MHEG Engine (Engine) is the central processing entity in the endsystem. It receives and interprets MHEG objects and controls the user interface. The Control Agent (CA) is the communication front-end for the networked MHEG Engine. The User Interface Agent (UIA) is responsible for the management of the autonomous Presentation Objects (POs). The POs realize audible, visible and interactive functionality. The management system comprises a Session Management Agent (SMA), an Authoring Agent, Security Agent, Locator Agent, Directory Agent, and Data Distribution Agent. The Stores/Video Server complex consists of MHEG Stores and Content Data Stores including audio and video data. Store and video server are responsible for content delivery of discrete or continuous monomedia.

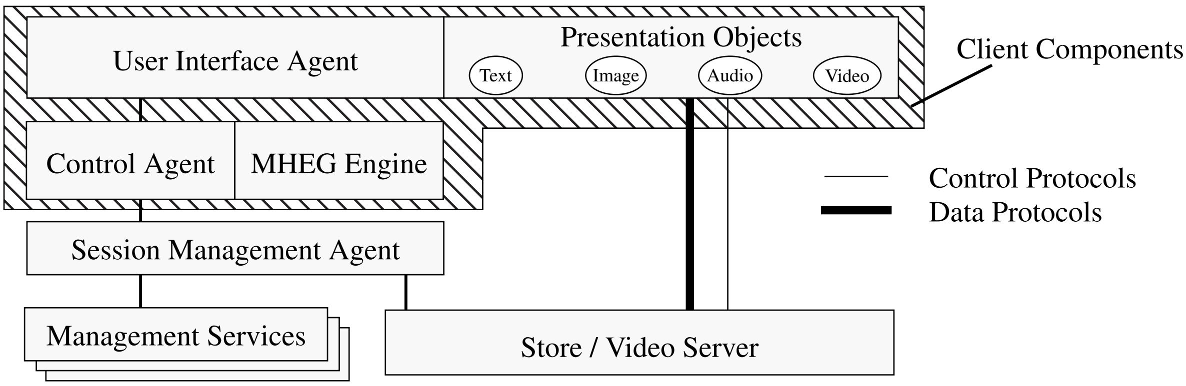

Fig. 3. GLASS System Architecture

4.1. Client Components

User Interface Agent and Presentation Objects

The UIA is responsible for the creation, control, and destruction of POs. The UIA communicates with the Engine by using a GLASS specific communication protocol. This protocol has primitives for session establishment, control and accounting, and primitives for PO control.

POs handle the presentation of content data objects and user interaction. They are responsible for presenting multimedia data to the GLASS system user. POs are created, modified and destroyed by the UIA on demand of the engine. To allow for concurrent processing of presentation tasks (e.g., 2D animation and video playback at the same time) UIA and POs are multithreaded. Atomic presentation elements of the GLASS system are JPEG images, MPEG-1 video/audio, and text.

As one of the key concepts of the MHEG standard, there are two phases for the presentation of contents: (1) preparing the content, and (2) presenting this content. For example, within phase 1 of the playback of a video the resource allocation (memory, CPU, filesystem) for the video server and the end-system, the connection set-up between video server and end-system, and the initialization of video decompression devices can be done. In phase 2, the video is shown by transferring the video data from the video server to the end-system and displaying it on the screen.

Communication between end-system and stores is done by separating the control and data flow into two network connections. The protocols for the control of monomedia data transmission is the Presentation Object Control Protocol (POCP), for data the Presentation Object Data Protocol (PODP) is used. POCP establishes and controls PODP which performs the actual data transmission. While POCP provides primitives such as “open data connection”, “start/stop data streaming”, “set stream speed” etc., PODP transmits raw data of different media types. The following media types are currently supported: video (MPEG-1), Audio (MPEG-1 audio and WAVE), Images (JPEG), and Text (plain text and a GLASS text format).

MHEG Engine

A special run-time engine is responsible for the execution of multimedia applications which in MHEG on the end system. This MHEG Engine manages elements of the user interface (represented by the POs) and the system resources, such as delivery and access networks. In the end system of GLASS, the MHEG Engine is the central processing entity. If additional MHEG objects are required (e.g., a new page is to be prefetch), requests are sent to the application server / management components. The Engine requests MHEG objects asynchronously from the Communication Agent (CA). The CA then contacts the MHEG store through the SMA. The result of this retrieval process is an asynchronous response as an input to the Engine. The process of interpreting MHEG objects is executed by different parts of the MHEG Engine, each of them fulfilling distinct responsibilities:

Obviously the interpreter plays a central role: It parses an object hierarchy that represents those MHEG Objects locally in the Engine that are under the interpretation process. The effect of one or more interpretation steps is typically a state transition of other objects. These state transitions are implemented in a way that reflects the semantics defined by MHEG. Events generated by user input or media synchronization, which are signaled from the PO to the UIA, do also cause state transitions. To enforce these state transitions, sending a request to other components might be necessary. Due to the asynchronous processing of these request, another component is needed for the MHEG Engine. This manages multiple tasks within one thread of control, that are being processed interleaved. The so-called scheduler component takes care of matching responses to requests, scheduling tasks, and managing context information.

A state transition in an object might also trigger an MHEG Link object. The triggered link is said to be ‘fired’. Therefore the Linkprocessor component needs to observe all links’ trigger condition and fire links. Firing of a link causes new objects to be sent through a FIFO queue maintained by the Scheduler to the interpreter for further interpretation. MHEG uses events rather then control structures to direct the flow of an application. This is a major difference between MHEG and the common scripting languages that applied a programming-language like flow model. Thus an efficient implementation of object linking is crucial for the overall system performance. Satisfying responsiveness, which basically depends on fast object linking, determined usability of the GLASS system.

The MHEG Engine needs to be implemented like an interpreter that dynamically acquires new MHEG objects and discards others, when they are not needed any more. Therefore the so-called object handling component of the MHEG Engine requests MHEG objects from the server, decodes them upon arrival, and removes objects from the interpretation process. An ASN.1 to C++ compiler, called SNACC, has been used to machine-generate the decoder for ASN.1’s Basic Encoding Rules (ASN.1 BER) from the MHEG syntax. This tool uses C++ classes to store the output of the BER decoder, which is also automatically generated. In the GLASS project it was decided to use this C++ object hierarchy as the local representation for MHEG Objects in the MHEG Engine. Though this choice gave implementation efforts a quick start, it proved to be unsatisfactory, because a machine-generated local representation is both highly inflexible towards changes in the evolving MHEG standard and it is also inadequate in terms of efficiency.

Figure 2 shows the interaction of the Engine components: User interaction events are passed to the Engine through the UIA and the CA. MHEG objects and events are processed by the Engine. Events and actions to be interpreted on MHEG objects are fed from neighboring components through the Scheduler into the Interpreter. Interpretation results in state transition which invoke requests to other system components and also trigger links. This may result in a request to the presentation systems or causes the retrieval of further MHEG objects. Actions result in the retrieval of new objects, or in the sending of requests to the presentation system.

Communication Agent

The CA handles the Engine communication with remote or local system components. In the session launch phase the CA establishes a connection to the SMA and initializes the Engine. It supports the Engine with provision of timer mechanisms and error

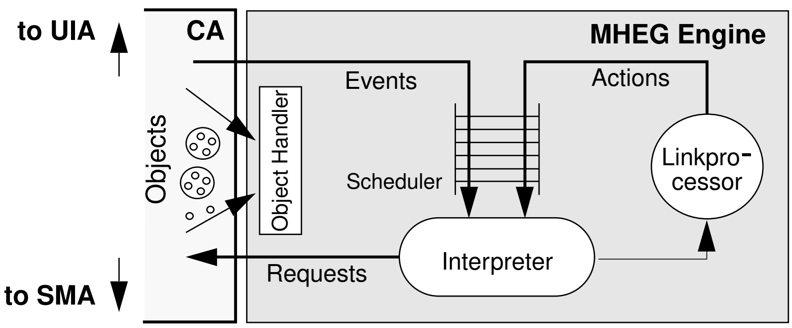

Fig. 4. : MHEG Engine Architecture

handling, provides abstraction from the communication protocols, and hides system dependencies from the engine. To test the stability of the Engine, it can also drop or generate events, insert delays or simulate neighbouring components.

The CA and the Engine can reside either in the end-system or in the application server. CA and Engine running on the end-systems provide better response time but require more local processing capacities. On the application server they reduce resource requirements in the end-system and centralize the interpretation service.

4.2. Server Components

The Server components can be split into two parts. The first part is the part specific to an MHEG session. It includes the session management, the MHEG object retrieval and the online generation of MHEG objects. This part is the Application Server. The second part is not specifically concerned with MHEG but provides the monomedia content. Of course, it interoperates with the Application Server. This includes provision of location information and querying for permission prior to content delivery to a requesting client. This part is a Data Store. The communication between Application Server and Data Stores is done by using the Store Control Protocol (SCP). This protocol contains primitives for the permission of data transfers between POs and Stores, for the retrieval of MHEG objects from MHEG Stores, and for accounting purposes.

Application Server

In the GLASS system, the tasks that are commonly associated with an Application Server are distributed among a number of components, which need not be located on the same machine. The SMA provides a common interface for its sub-agents, the Locator Agent, the Accounting Agent, and the Security Agent. All tasks that do not require dynamic generation of data are solved by these sub-agents alone. In detail, they handle:

- session setup,
- user authentication,
- location resolution for content data, and
- accounting for MHEG objects.

Furthermore, it implements the retrieval of MHEG objects from MHEG stores. In contrast to content data that is referenced by MHEG objects, not only references to the objects are sent to the CA on request, but the objects themselves are retrieved from their respective MHEG stores or gateways and sent in response.

The MHEG stores are simple servers that return MHEG objects to the SMA on request. These objects have been generated offline. The objects have been loaded into the store, and references to the objects have been added to the Locator Agent's databases.

Gateways

The gateways implement the full functionality that is expected of an Application server. They generate both monomedia content and multimedia structure information on the fly and provide it to the SMA for retrieval. Within the GLASS project, a number of gateways have been implemented to provide access to services external to the project. These include:

- TV/Radio gateway
- Fax gateway
- WWW gateway
- Multimedia Mail gateway
- BTX gateway

and a number of small service application that test specific engine features, implement a server-driven clock, implement an Xbiff-like service that informs the user about incoming mail etc.

These gateways can be split into two groups. The first group is focused on content conversion and has only a fixed MHEG presentation which is used as a framework. The only MHEG objects that are actually generated on the fly are content objects whose references to the actual content are updated according to the generated content data. This group comprises the TV/Radio gateway and the Fax gateway. The other group focuses on the generation of MHEG objects on the fly. Content is also converted and made available for retrieval, but maintaining reference consistency throughout the application and making MHEG objects available when they are required by the engine is the main problem in this second group of gateways.

The gateways of the second group are based on generation libraries that simplified the creation of MHEG objects. However, the other problems of gateway implementations had to be solved individually because of different access systems, data formats, and behaviour of the integrated services.

For example, the major complexity of the gateway to the World Wide Web lies in detecting and processing each hyperlink of a retrieved HTML page and in splitting a page that is bigger than the screen. Each hyperlink must be converted to a new object request action within the MHEG presentation, and it must be marked uniquely. This way, when the user clicks on the respective place in the MHEG presentation of the Web page, it enables the gateway to map the object request to the URL of the original hyperlink and to retrieve and process the next Web page.

BTX is a popular German online service which has recently been renamed to “Telekom Online”. For the BTX gateway, on the contrary, the major complexity is in processing data that is provided by the BTX host without explicit request of the client or the presentation. One example for unrequested data sent to the gateway by the BTX host is a credit timer that counts the time that is spent on a specific page. Though this application may not exist on any BTX page at all, the Cept standard allows this kind of host-driver display update. In a situation where the BTX host initiates an update, the gateway must trigger an event in the Engine. This is required because in an MHEG system, the client is the driving instance, while in a BTX system, the client is only a dumb character-oriented terminal and the BTX host is the driving instance. The gateway must implement a means to translate between these two concepts.

Data Server

The Data server consists of two entities, the Store and the Video Server. The store is responsible for the storage of content data and their transmission to the end-systems. The store provides functions for controlling the exchange of content data with POs, and transport of continuous-media data.

The video-on-demand server is designed for IBM RISC System/6000 workstations running under IBMs AIX Version 3 operating system.[1] The video server provides mechanisms for the dynamic addition of hardware- and protocol-specific stream handlers. It supports the guaranteed delivery of high quality, time-critical continuous-media data such as video, audio, and animation. The video server allows for the delivery of *streams* of multimedia data (e.g. video) with guaranteed quality-of-service because it includes resource management functionality for network and local system resources such as CPU and file systems. Additional store functions have already been designed: an accounting information interface to the SMA, a communication interface to Locator Agent and Data Distribution Agent.

5. EXAMPLE APPLICATIONS

A typical iTV navigation scenario is shown in Figure 2. The scenario consists of a services home page which presents the service providers registered in the iTV system, and the home page of a service provider. Both pages are composites of one background image and small buttons. Buttons are implemented by images which are selectable with the user interaction devices. Buttons usually represent effects like the transition to other directories of service gateways. If the button labelled with “GLASS” is pressed in the selection menu an audio feedback is played and the GLASS service provider home page is presented.

Typical application of an interactive television system have been implemented: video/audio on demand, home shopping, multimedia information retrieval services, application gateways to multimedia mail/WWW/BTX. and delayed broadcast.

6. COMPARISION OF DAVIC AND GLASS

The Digital Audio-Visual Council (DAVIC) [1] is one of one of the most important system standards for digital and interactive multimedia and television systems. The purpose of DAVIC is to favour the success of globally networked multimedia systems

1. Within the GLASS project a DEC Alpha / DEC UNIX based data server using a different architecture has been built as well.

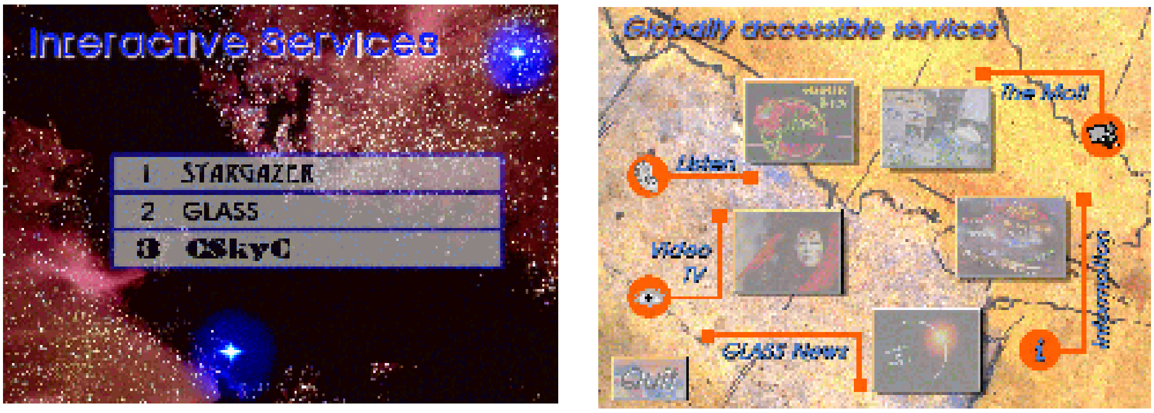

Fig. 5. : Navigation Service Using GLASS

by specifying and opening the interfaces of the content provider, service provider, network provider and client systems. The GLASS team is involved in the DAVIC standardization group and contributed to the DAVIC application, set-top unit, server, and technology technical committees.

Table 1: Comparision of DAVIC and GLASS

Component	DAVIC	GLASS
Information Representation		
- multimedia format	MHEG-5	MHEG-1
- service information	DVB SI	not implemented
- monomedia formats (video,audio,graphics,text)	MPEG-2, MPEG-1[a], CLUT[b], Unicode	MPEG-1, MPEG-1, JPEG, Text[c]
- multiplex	MPEG-2	MPEG-1
- content packaging format	to be defined	not implemented
Session Management		
	DSM-CC (U-N)	SMCP[d]
Client protocols to access		
- video server	DSM-CC stream commands	POCP
- application server	DSM-CC file commands	SMCP[e]
Transport Protocols		
- delivery network	AAL5, MPEG-2 TS[f]	TCP/IP over ATM, Ethernet ..
- access network	TCP/IP over ATM, ...	TCP/IP over ATM, Ethernet ..
System Management		
	SNMP MIB / SNMP	GLASS specific protocols

a. The definition of a format for linear audio is under consideration.
b. Additionally, the MPEG-2 still picture format has been defined as graphics format.
c. GLASS defined an own text format.
d. Session Management Control Protocol; GLASS proprietary
e. MHEG object request/response mechanisms

f. The following types of delivery networks are defined by DAVIC: ADSL, FTTC, HFC (non-ATM, ATM), Active Network Termination Delivery (ATM), ISDN, Herzian Satellite Network, VDSL, PSTN

In the above table GLASS is compared with the components and protocols of the DAVIC system stnadard. It can be seen that the following work items remain for the next releases of GLASS which is aimed to comply to the DAVIC system standard: (1) MHEG-5 compliance, (2) DSM-CC stream and file command, (3) DSM-CC User-to-Network signalling, (4) MPEG-2 video and transpart streams, (5) management based on SNMP and a standardized set-top unit/server MIB, (6) support for DAVIC delivery systems (e.g., AAL5).

7. CONCLUSION

Interoperability and reusability is one of the key issues for the success of digital and interactive television systems. This paper introduced the MHEG standard and its importance for interoperable application programming. To our knowledge GLASS is one of the most advanced distributed multimedia systems based on MHEG. The experiences gained in the GLASS project show that interoperability at the application level can be reached by using run-time engines which are MHEG standard compliant. The results of the GLASS project have fundamentally influenced the standardization of MHEG-1, MHEG-5, and the *Digital Audio Visual Council (DAVIC)* — a consortium of 150 companies including the leaders of the consumer electronics, computer, network provider, service provider and content provider market. DAVIC chose the MHEG-5 format for the higher layer set-top unit application programming interface and as multimedia information representation format.

8. ACKNOWLEDGEMENTS

This work has been supported and partly sponsored by the DeTeBerkom. The authors thank all partners of the GLASS consortium, namely DEC CEC, DeTeBerkom, GMD Fokus, Grundig Multimedia Solutions, TU Berlin PRZ for the co-operative work. The authors express their gratitude to Hans Werner Bitzer, Thorsten Illies, Stefan Koenig, Thomas Meyer-Boudnik, and Olaf Rehders for their discussions and contributions.

9. REFERENCES

1. L.Chiariglione: *DAVIC — Preparing technology for end-to-end interoperability.* IEEE Multimedia Newsletter, December 1995.
2. Digital Audio-Visual Council (DAVIC): *DAVIC1.0 Specification Revision 4.1.* September 1995.
3. DEC, GMD Fokus, Grundig Multimedia Solutions, IBM ENC, Technical University of Berlin PRZ: *BERKOM Globally Accessible Services: System Specification 1.0.* BERKOM, Berlin, May 1994.
4. ISO/IEC IS 10918:1992: *Information Technology – Digital Compression and Coding of Continuous-Tone Still Images* (*JPEG*). 1992.
5. ISO/IEC IS 11172:1992: *Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s (MPEG).* 1992.
6. ISO/IEC IS 11544:1992: *Information Technology – Digital Compression and Coding of Bi-level Images (JBIG).* 1992.
7. ISO/IEC WD 13818-6:1994: *Information Technology – Coded Representation of Multimedia and Hypermedia Information Objects (MHEG) – Part 1: Base Notation (ASN.1).* June 1993.
8. ISO/IEC CD 13552-1:1993: *Information Technology – MPEG-2 Digital Storage Media Command and Control Extension (DSM-CC).* November 1994.
9. ITU-T Draft Recommondation T.170: *Audiovisual Interactive (AVI) Systems – General Introduction, Principles, Concepts and Models.* Second Revision, Geneva, CH, 16-25 November 1993.
10. T. Meyer-Boudnik, W. Effelsberg: *MHEG - An Interchange Format for Interactive Multimedia Presentations.* Accepted for IEEE Multimedia Magazine, 1995.