Experiments with MHEG Player/Studio : An Interactive Hypermedia Visualization and Authoring System

Seungtaek Oh, Yung Yi, Seunghoon Jeong, and Yanghee Choi
Department of Computer Engineering, Seoul National University,
Shinlim-dong, Kwanak-goo, Seoul, Korea
Tel: +82-2-874-0286
Fax: +82-2-876-7171
{ ducky, yiyung, shjeong }@mmlab.snu.ac.kr , yhchoi@smart.snu.ac.kr

Younghwa Ko
Samsung Data System
Tel: +82-2-3429-3483
yhko@samsung.co.kr

Abstract

With the growing needs of information sharing and exchange, MHEG-6(Multimedia Hypermedia information coding Expert Group - part 6) standard is defined so as to provide the international standard for representing common coded interactive multimedia information, presenting and exchanging information using the same syntax and semantics in a heterogeneous environment. We developed a prototype system for visualizing and authoring hypermedia information based on MHEG-6 coded representation. The MHEGPlayer is a visualizing system designed to satisfy the various system requirements of the MHEG-6 and DAVIC(Digital Audio Video Council) standards, including support for various time-critical complicated synchronizations and interactions between MHEG objects on a MHEG-5 Engine and execution elements on a Virtual Machine. The MHEGStudio is a user-friendly interactive integrated authoring system designed to support convenient and simple operations without requiring overcomplicated standard-specific details. In this paper, we comment on the standard and the overall architecture of the MHEG Player/Studio. A scheme to use MHEG-6 in the WWW(World-Wide-Web) environment is suggested.

Keywords :

MHEG, Multimedia, Hypermedia, Coded Representation, Presentation System, Authoring System

Topic Area : MULTIMEDIA AND TELECOMMUNICATIONS

Experiments with MHEG Player/Studio : An Interactive Hypermedia Visualization and Authoring System

Abstract

With the growing needs of information sharing and exchange, MHEG-6(Multimedia Hypermedia information coding Expert Group - part 6) standard is defined so as to provide the international standard for representing common coded interactive multimedia information, presenting and exchanging information using the same syntax and semantics in a heterogeneous environment. We developed a prototype system for visualizing and authoring hypermedia information based on MHEG-6 coded representation. The MHEGPlayer is a visualizing system designed to satisfy the various system requirements of the MHEG-6 and DAVIC(Digital Audio Video Council) standards, including support for various time-critical complicated synchronizations and interactions between MHEG objects on a MHEG-5 Engine and execution elements on a Virtual Machine. The MHEGStudio is a user-friendly interactive integrated authoring system designed to support convenient and simple operations without requiring overcomplicated standard-specific details. In this paper, we comment on the standard and the overall architecture of the MHEG Player/Studio. A scheme to use MHEG-6 in the WWW(World-Wide-Web) environment is suggested.

Keywords

MHEG, Multimedia, Hypermedia, Coded Representation, Presentation System, Authoring System

I. Introduction

The progress in the development of high-speed networks and real-time multimedia information processing technologies has brought a new generation of multimedia services. To realize such services, certain issues require resolution such as Open Systems in a networked environment, the facilitation of information sharing among homogeneous systems, the provision of user-friendly environments, etc. In the case of information sharing, a substantial amount of research has been carried out on various levels and in different environments. However, with regard to the multimedia field, the prevalence of vendor-driven markets has prevents from sharing various media information among different vendors with different purposes. Furthermore, standards have been almost non-existent for authoring, storage, network transfers, and presentation of complex interactive multimedia representations on different systems.

With the rise of multimedia in the 1990s, the process of standardization has begun for multimedia services, multimedia user system, network service architectures, methods for coding and representing multimedia information, etc [4][5][6][7][8][9]. The selection of the MHEG standard as the coded representation scheme for interactive multimedia information has brought the provision of user-level services by DAVIC-supported systems closer to full realization. The MHEG standard is applied in the architecture for exchange and representation of multimedia information among end systems [Fig. 1]. The MHEG-5 standard defines the common coded representations used in interactive multimedia applications to support information sharing in heterogeneous environments. The MHEG-6 standard defines supplementary high-level computation functionality over MHEG-5 applications using the Java Virtual Machine mechanism.

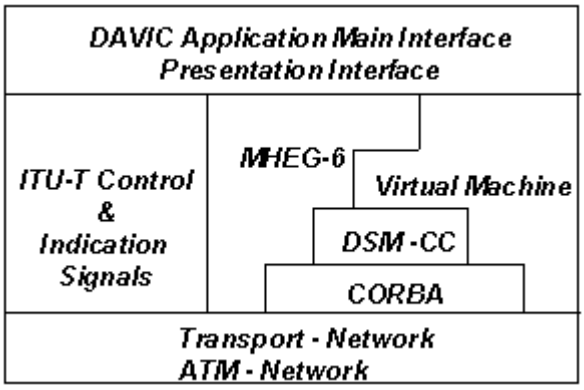

Fig. 1. DAVIC architecture

The designs of the MHEGPlayer, a DAVIC-supported presentation system, and the MHEGStudio for authoring of MHEG-6 documents, are laid out and developed in this paper. To represent complex interactive multimedia information, MHEG documents are constructed through compound consolidation of basic components, and are presented using event-driven interpreting system. The client system MHEGPlayer solves various problems arising from the real-time requirements of multimedia, the need to schedule synchronous/asynchronous events and processing tasks, and the complex interaction between MHEG-5 RTE(run-time engine) and Java VM(Virtual Machine). Due to the difficulties inherent in constructing complex interactive multimedia documents through basic component aggregation and using complicated referencing mechanisms, as well as a number of invisible components: MHEG documents are very hard for the average user to understand and modify as well as to create. The MHEGStudio is a user-friendly visual authoring system that resolves such problems. An overview of the MHEG standard is introduced in chapter 2. In chapters 3 and 4, the architecture and key functionalities of each system are described. Chapter 5 concludes this paper with a suggestion for using MHEG-6 on WWW environments.

II. MHEG Standards

MHEG-5 is the standard defined by ISO/IEC JTC1/SC29/WG12 which defines how a final-formed coded representation of interactive multimedia and hypermedia information can be exchanged freely in a client/server environment across heterogeneous platforms from different vendors [1]. The formal specification of all data structures is described in ASN.1, the so-called MHEG-5 classes [3]. The semantics of these MHEG-5 classes define the functionality and requirements for an MHEG runtime environment. The MHEG-5 object is an instance of a MHEG-5 class and a meaningful set of objects is called a MHEG-5 application or MHEG-5 document.
Once any MHEG application is written, it can be presented to any user on any platform supporting the MHEG-5 standard. The MHEG-5 Engine is a presentation subsystem that can understand and present correctly MHEG-5 application. A MHEG object can be stored in a server system, downloaded to client, dynamically interpreted and displayed to a user on a MHEG-5 Engine [Fig. 2].

A MHEG-5 application consists of one Application and at least one Scene class objects. In general case, the MHEG Application object is just a startup of an application and the MHEG Scene object represents a scene to be presented to a user. A Scene object describes overall properties of a scene such as the coordinate system. Also, it includes Ingredient objects as the component objects of different functions.

To represent media contents, several kinds of audio/visual classes such as Bitmap, Audio, Video and Stream, are defined. These classes also provide the encapsulation of media contents. To represent graphical user-interaction, some kinds of button-styled object classes are defined. To provide basic arithmetic, there is some kinds of Variable classes and primary operations such as add, modulo. To decorate GUI(graphical user interface), an application can make use of CursorShape, Font and Palette class objects. Furthermore, the Link object provides the event-driven conditional synchronization. By use of this, an application can describe plentiful synchronization scenarios. The event can be generated from a user interaction, a timer-expiration, a change of the state of an object, etc. The Action described in a Link object represents what to be done when the link condition is triggered by events. With these describing power, MHEG-5 documents is more than just a anchor-based hypermedia : A number of graphical effects and spatial-temporal dynamic behaviors can be described.

The characteristics and benefits of MHEG-5 are as follows : the encapsulation of media contents by ingredient objects, the composition of such content-related objects and objects directing how to interact with the user and system events, the presentation control of bitmaps, sounds, videos, buttons, scrolls, etc. by control of object attributes, and the spatial/temporal dynamic conditional synchronization with relations and states of objects.

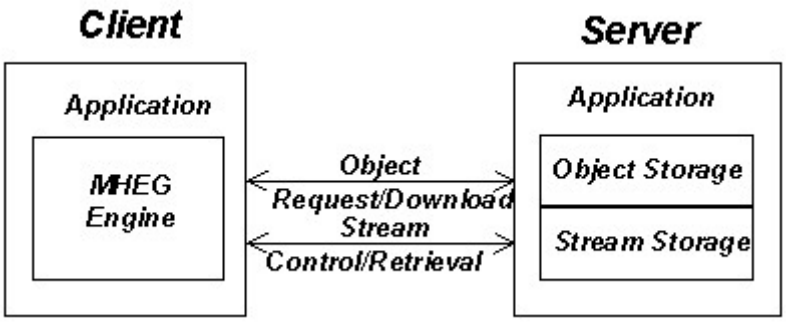

Fig. 2. Client/Server environment using MHEG

A MHEG-5 application can describe various kinds of GUI-related information and useful functions for the manipulation and display of multimedia information, but its computational power is too weak to describe more

complex applications. For example, because such application can include only the StringVariable and IntegerVariable objects which can handle only basic operations, a simple application such as a calculator application cannot be written. Moreover, to provide certain critical business services like homebanking, various encryption/decryption mechanisms like RSA should be used. The MHEG-6 standard has been introduced with a view of addressing these needs[2]. MHEG-6 provides not only such computational power but also a complex scripting mechanism based on Java VM(Virtual Machine). With these added features, the MHEG-6 can thus be viewed as a superset of MHEG-5. The MHEG-6 Engine is a presentation system supporting MHEG-6 standard. The details are described in next chapter..

III. MHEGPlayer

The MHEGPlayer is a client-side implementation of MHEG-6 Engine. The architecture of the MHEG-6 Engine is shown in [Fig. 3]. The MHEG-6 Engine consists of a MHEG-5 Engine, a Java VM, a ClassMapper and a MHEG-5 API. Downloaded MHEG-6 applications can be viewed as two parts : the MHEG-5 application and the IP class codes. The MHEG RTE decodes the application, sends the MHEG-5 application part to the MHEG-5 RTE and loads class codes to a Java VM via a ClassMapper. The MHEG-5 RTE subsystem interprets and displays the MHEG-5 application. The Java VM executes the IP object class codes. A MHEG-5 API interchanges any method call and results between the MHEG-5 RTE and the Java VM in a manner similar to RPC implementation. Within codes in a Java VM, an access to a wrapper object is matched to an access to a corresponding MHEG-5 object in MHEG-5 RTE and this matching is done through the MHEG-5 API using a system-dependent message system. In [Fig. 3], the architecture of the MHEG-6 Engine is illustrated.

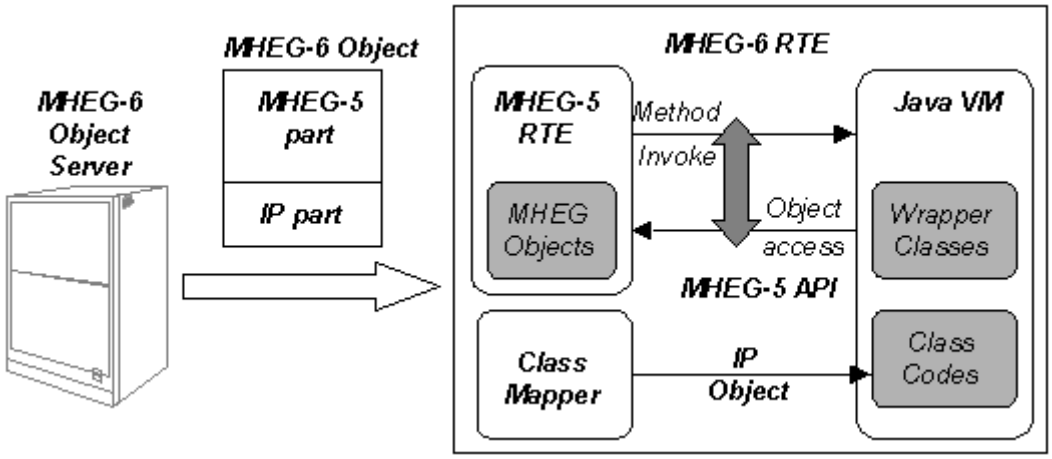

Fig. 3. Architecture of a MHEG-6 RTE

MHEG-5 Run Time Engine

MHEG objects are stored in ASN.1 BER(Basic Encoding Rule) format. To provide a given service, a MHEG RTE downloads the appropriate Application and Scene objects, then decodes those into internal data structures. The MHEG RTE also handles user interactions and internal events to dynamically interpret and present MHEG objects. To provide visual hierarchy, RTE should contain a internal windowing system with, among others, window boundary, clipping, and z-order stack. To provide interactivity, RTE should gather physical UI(user interface) events and transform those to MHEG events. A MHEG-5 Engine consists of the following functional subsystems. [Fig. 4] shows the architecture of a MHEG-5 RTE.

- ***User Interface subsystem*** : processes user-interaction system events and transforms these system messages to MHEG events. These events should be processed asynchronously, so that events are put in an asynchronous event queue of an event scheduler in the Link processor subsystem. For example, if a user pushes a button on display, the UI subsystem recognizes this and calculates the appropriate display position, transforms into a UserInput event with position data, and puts this event on an asynchronous event queue in an event scheduler. The main role of the UI subsystem is recognition and transformation of system-dependent user interactions.
- ***Presentation subsystem*** : processes media-oriented materials. It manages an internal windowing subsystem and makes use of its own media-dependent decoders/displayers. For example, it provides generic windowing system functions and MPEG decoder, so that a MPEG video stream is displayed on an overlapped window. It also provides the rendering functions for buttons, bitmaps, captions, line-arts, etc.
- ***Object Handler subsystem*** : manages the state of objects and provides access points to these objects. Other

subsystem can use this subsystem to access and manipulate an object. The Object Handler retrieves objects and decodes them into internal data structures. Also, all methods applicable to objects are implemented in this subsystem using object-oriented techniques with efficiency.

- ***Link processor subsystem*** : processes events, interprets overall object states and link conditions, and applies appropriate actions. In the Event scheduler, it should find the most appropriate event to be processed, with taking into account various priorities and context-sensitive meaning. For example, synchronous events are processed with a higher priority than asynchronous events, events of same kinds are processed in FIFO manner, and events created during processing of the first event are processed with a higher priority than the other events pre-created after the first event. The management of the multilevel priority event queues is intended to solve this problem. The Link process subsystem compares a current event to link conditions described in active Link objects, then extracts elementary actions to be executed accordingly. These actions are inserted into the Action queue of Action scheduler. The Action scheduler operates so as to assure that the various behaviors of actions should maintain consistency with overall RTE semantics. For example, an execution of an action brings out new events and actions, but these new actions should be executed immediately in some cases and not immediately in dependence of RTE semantics. The need for such complex context-sensitive interpretation necessitates such a complicated Link processor mechanism.

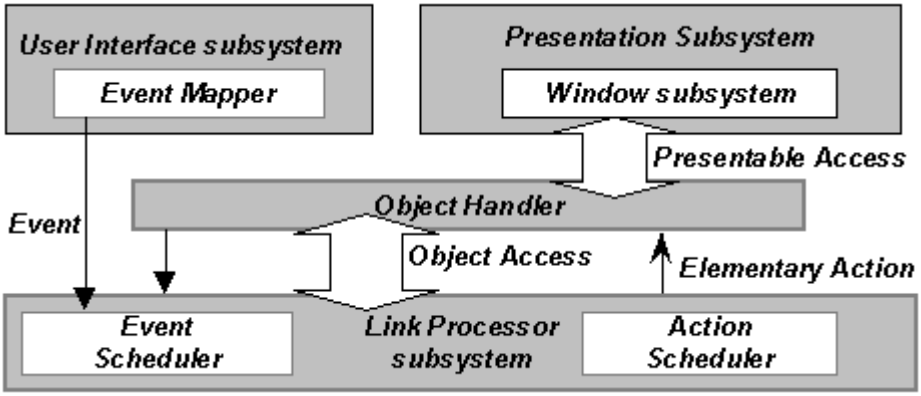

Fig. 4. Architecture of a MHEG-5 RTE

MHEG-6 Run Time Engine

With MHEG-6, we can control mathematical functions or a hundred number of variables with ease. All we have to do is to make the external Java class code according to the VM class format using MHEG-5 API class package. This external code requires little effort using high-level Java languages such as C or C++. This feature provides a surprising enhancement to the MHEG service in addition to the light-weighted MHEG engine. The size and complexity of MHEG application could be dramatically decreased. The MHEG-6 extension supplies the environment with a running Java VM. In addition to a Java VM, MHEG-6 engine needs the Kernel API and MHEG-5 API. The Kernel API is the set of classes supplying the available basic Java bytecodes running on top of Java VM.

The MHEG-5 API is used by running class codes to access the attributes of a MHEG object dynamically or call the elementary action of a MHEG object in the MHEG RTE. Any MHEG-5 elementary action corresponds to a MHEG-5 API Java method. The MHEG-5 API supplies elementary actions for only presentation function and retrieval of attributes. Asynchronous message mechanism is used for transmitting information between the RTE and the Java VM. This message contains information of a desired action, an error value, a target object, a result value and arguments for the action. When the Java VM needs to transfer the information to the RTE, it sends the asynchronous message, waits for the result and process the result of RTE. When MHEG RTE needs to invoke Call action, RTE process the action using this asynchronous mechanism synchronously. In case of Fork action, RTE processes the action using this asynchronous mechanism asynchronously.

Another requirement of the MHEG-6 engine is location-independent Java class loader. The default class loader of a Java VM can load only the classes that reside in local file system. But, MHEG-6 requires that the class loader can load the classes in other location other than the local file system. Precisely, MHEG-6 Engine can extract the class codes from IP object and the class loader, the ClassMapper, loads this class codes. To maximize efficiency, the class loader maintains a hashing table for classes, so that before the class loader tries to load a class, it can check whether the class is already registered. If already registered, the class code is retrieved simply using the reference information of a corresponding IP object saved in the table without searches for class codes. If not registered, the request can be determined as an illegal request without full searches for codes.

A MHEG-6 Engine can show the end-user all the MHEG-5 application and defines the structures of

InterchangedProgram object and a method of availability. After that, it launches the Java VM. In order for a Java VM to load and run class codes given by the MHEG-6 Engine, not only MHEG Engine initiates to load class codes of a IP object, but also Java VM can resolve the required class codes which is retrieved from MHEG Engine. The ClassMapper plays a role in this bidirectional mapping. Furthermore, the MHEG-6 engine must support the bidirectional invocation and return mechanisms through MHEG-5 API, so that the MHEG-5 Engine can invoke a static main method of a loaded class in Java VM, and the running code in Java VM can access the MHEG object in MHEG-5 Engine. These operational sequences are illustrated in [Fig. 5].

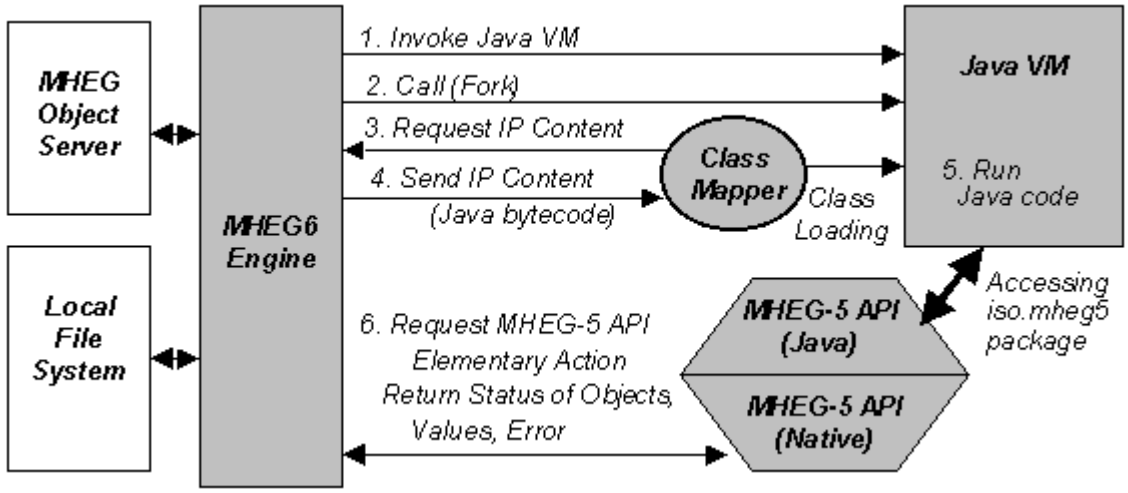

Fig. 5. Operational Sequences of MHEG-6 RTE

In the flow architecture above, the important components are the Java VM, the MHEG-5 API running on the Java VM, the class loader and the RTE. The detailed sequence of method invocation is illustrated in [Fig. 6].

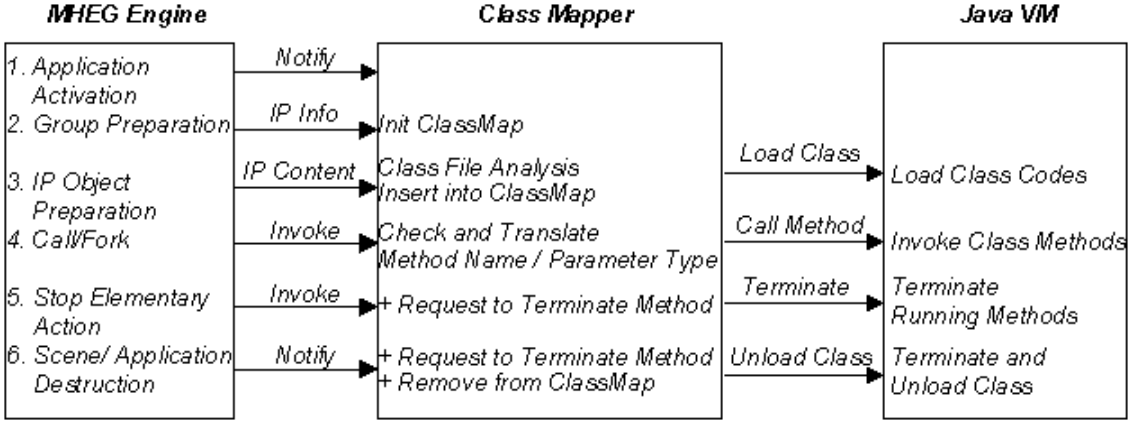

Fig. 6. Interworking of MHEG-5 Engine, ClassMapper and Java VM

IV. MHEGStudio

To assure wide use of any specific representation scheme of multimedia information, a user-friendly authoring tool has been developed. In the case of HTML, in spite of its complex format to directly create and modify, it has been widely used among general users following the appearance of HTML authoring tools. Compared with HTML, the MHEG format is highly complex, making it hard for users to directly edit MHEG documents. The MHEGStudio is a user-friendly, easy-to-use MHEG authoring system [Fig. 7].
[Fig. 8] illustrates the architecture of MHEGStudio and the key functionalities are as follows :

- ***Hierarchical Management*** - Project & Resource : MHEGStudio provides a project management scheme, so that a user can easily manage application hierarchies. Each project is stored individually in a structured file format and it can be accessed and modified at any time. A project is structured so that the user can see it as a tree view, and reuse included objects and contents in other projects with ease. Furthermore, a resource database is included in the project so that any media contents can be used simply as resources.
- ***Hiding of details*** : object reference and content reference mechanisms are very hard for the user to recognize. Unlike URL, an object in a group is referenced by a unique object number. But, sometimes the

user may want to name an object with a meaningful string. For example, a pushbutton labeled *"Cancel"* can be named for and memorized as *"CancelButton"* instead of a number 30997 by the user. The MHEGStudio manages an object name (symbol) database for every project, so the user can define object name and the MHEGStudio can assign an appropriate free object number, or vice versa. Any complex referencing mechanisms are hidden to end-users. For detailed modification, the Object Property Sheet is provided for the easy creation and modification of any object, as in a public rapid development system.

- ***Easy use of dynamic synchronization*** : the main problem is that it is too hard to represent and understand an event-driven dynamic synchronization in the MHEG manner. To solve this problem, MHEGStudio hides Link objects and any conditional link is shown to the user as an event-method of a target object. For example, to describe that *"if bitmap A is available, then show it"*, there may be a Link object describing that *"the source is the object A, link condition is availability of object A, target is object A, action is activation of object A"*. For each event to be processed, at least one Link object is needed, making it is hard to understand and keep track of such a number of Link objects. In the case of the MHEGStudio, the user only needs to click on Bitmap A displayed on a virtual Scene window, registers the method named *"OnAvailable"*, and then select *"Activate"*. For each action, MHEGStudio shows associated events and related actions so that the user can keep track of the presentation scenario easily.
- ***WYSIWYG editor*** : any object can be easily created in a visual manner. To create a new project, a click on *New Project* button is all that is needed. An application object is automatically generated and the user can create a scene from the scene templates. To create a bitmap object, the user selects the *"CreateBitmap"* button and drag-drops the preferred region on a virtual scene window. All objects are shown as a window or an icon. A double click on this visualized object opens the Object Property Sheet for that object. The MHEGStudio supports Online Help System for the user to understand, use correctly the complex attributes.
- ***Templates for complicated interaction*** : MHEGStudio adopts template scheme to provide easy creation of scenes, lists, etc. For example, if the user wants to create a dialog with two button named *"OK"* and *"Cancel"*, the user selects a Dialog template with OK and Cancel button in templates, then the MHEGStudio creates related objects such as bitmaps, hotspots, pushbuttons, links to process traditional dialog behavior. Various menu systems can be built in this manner.
- ***Virtual Execution Environment*** : Just-in-time compilation and execution makes possible the rapid detection of incorrect authorizations and enables the immediate reactions. Inclusion of a Java IDE and a MHEG-6 Engine provides an immediate virtual executing environment.

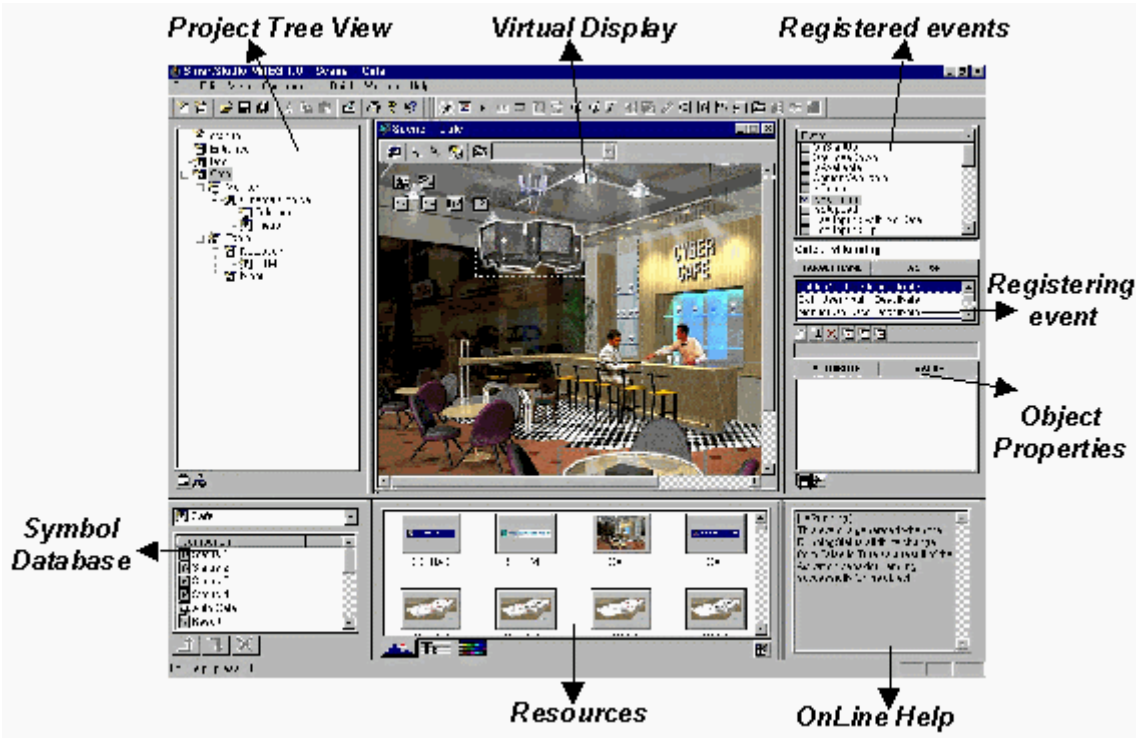

Fig. 7. A Screen Shot of MHEGStudio - Authoring Cyber Cafe

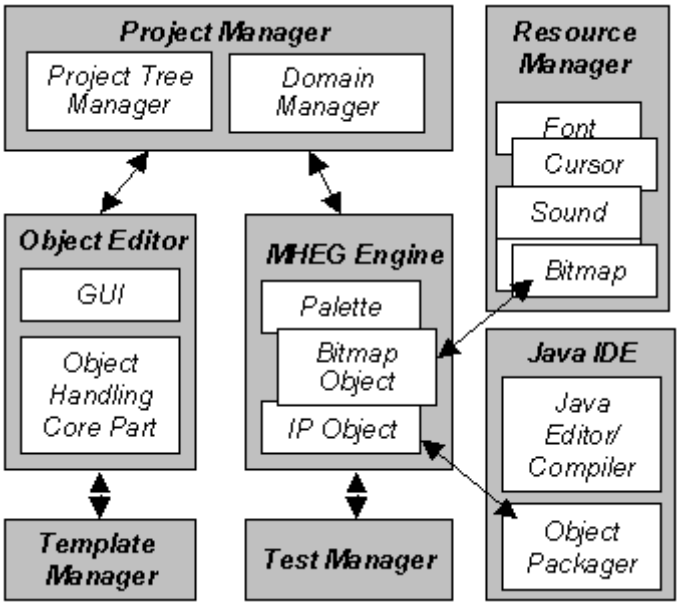

Fig. 8. Architecture of the MHEGStudio

V. Conclusion

The exchange, presentation and sharing of multimedia documents require standardized formats not only for monomedia contents, but also for dynamic behaviors and structures of complex interactive applications[10]. The MHEG-5,6 standard could provide these formats, which are available and could eventually see popular use.

This project aims to develop a prototyped DAVIC-compliant client system and accompanying authoring system, under the sponsorship of Samsung Data System. In this paper, we discussed the MHEGPlayer, a presentation system supporting MHEG-6 coded representation. Also, we introduced the MHEGStudio, a user-friendly interactive authoring system. We considered the requirements, functionalities, and architectures of these two systems. They are expected to be further developed and released by Samsung Data System as SmartPlayer and SmartStudio.

To accelerate the use of MHEG standards, we suggest a way of using MHEG in a WWW environment that does not require significant change. To achieve this, the suggested object/media retrieval scheme in DAVIC, DSM-CC and MPEG-2 TS, could be replaced by HTTP, RTP and RTCP protocols. Furthermore, some relevant object referencing mechanisms should be changed to reflect the appropriate URL formats. To achieve this, HTTP message formats are further defined. Web servers should handle not only HTTP but also RTP and RTCP for Stream objects with a session control preserving the context of MHEG Domain and providing VCR-like functions. Also, MHEG-6 Engine should be integrated into Web browsers as a MHEG document viewer(or plug-in) supporting such protocols. We expect to see such implementations soon.

References

[1] ISO/IEC IS 13522-5:1997, "Information Technology Coded Representation of Multimedia and Hypermedia Information Objects (MHEG) Part 5 : Support for Base Level Interactive Applications."

[2] ISO/IEC DIS 13522-6:1997, "Information Technology Coded Representation of Multimedia and Hypermedia Information Objects (MHEG) Part 6 : Support for Enhanced Interactive Applications."

[3] ISO/IEC IS 8824:1987, "Information Processing Systems Open Systems Interconnection Specification of Abstract Syntax Notation One (ASN.1)."

[4] ISO/IEC IS 10918:1992, "Information Technology Digital Compression and Coding of Continuous-Tone Still Images (JPEG)."

[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)."

[6] F. Kretz and F.Colaitis, "Standardizing Hypermedia Information Objects," *IEEE Comm. Magazine*, Vol.1, No. 5, May 1992.

[7] R. Price, "MHEG: An Introduction to the Future International Standard for Hypermedia Interchange," *Proc. 1st ACM Int'l Conf. on Multimedia*, ACM Press, New York, 1993..

[8] N.S. Borenstein, "MIME: A Portable and Robust Multimedia Format for Internet Mail," *ACM Multimedia Systems*, Vol.1, Springer, New York, 1993.

[9] J.-J. Sung et al., "Hypermedia Information Retrieval System Using MHEG Coded Representation in a Networked Environment," *Proc. 2nd Int'l Workshop on Multimedia: Advanced Teleservices and High-speed Comm. Architectures*, R. Steinmetz, ed., Springer Lecture Notes in Computer Science, Vol. 868, Berlin, 1994, pp. 53-66.

[10] B. Markey, "Emerging Hypermedia Standards: Hypermedia Marketplace Prepares for HyTime and MHEG," (U.S.) Assistant Secretary of Defense (Production and Logistics), Washington, D.C., PB92-120328, 1991.