

Experiences with the Development of an MPEG-4 oriented PC Multimedia Application

Thomas Stingl, Rico Dreier, Olaf Barheine^a

Forschungszentrum Informatik Karlsruhe^b

ABSTRACT

MPEG-4 is meant to be the integrating standard concerning Multimedia in the future. This paper wants to highlight experiences from the participation in the EU project EMPHASIS, the authors within a Research Institution being involved mainly in application definition and system integration. Based on this a general framework for system design comparison is outlined.

An application scenario for an MPEG-4 Demonstrator was specified to highlight both the benefits and features of MPEG-4 - based on Object orientation and Interaction, slogan: "Objects on demand" - and the outcomings of the specialised components work within the project.

Two different approaches for system integration were taken during the project term, one being a flexible combination of a Java System framework with native Decoder-software, the other a more MPEG-4 phase 1 related C/C++ Implementation based on first Standard drafts and the official Reference Software, but with an unique extension related to the usage of dedicated European multimedia accelerator hardware. Both demonstrators are described and compared with their advantages and disadvantages.

Further on, coming from the background of Formal System Design and CASE, an approach is described to bridge the gap between design and realization of control systems like e.g. in the aerospace sector, and the new possibilities of software-based, distributed multimedia systems based on the experiences described above. As these application areas will overlap in near future, a common view and computer supported comparison of potential methods and standards is needed; the outline and first experiences with such a system will conclude the presentation.

Keywords: MPEG-4, Multimedia, System Design, CASE

1. INTRODUCTION

MPEG-4 is an ISO/IEC standard being developed by MPEG (Moving Picture Experts Group), the committee which also developed the standards known as MPEG-1 and MPEG-2. MPEG-4, whose formal ISO/IEC designation will be ISO/IEC 14496, will be released in November 1998 and become an International Standard in January 1999. For further general information we refer to the available official information².

At about the same period in time when MPEG-4 was initiated, the ACTS programme launched several research projects dealing with this new standards, one of them AC 105 EMPHASIS¹.

EMPHASIS started in October 1995 and ended in September 1998. The main goals were to support via European industry and research institutes the MPEG-4 efforts both concerning standardization and implementation.

^a E-mail: stingl@fzi.de, rdreier@fzi.de, barheine@fzi.de

^b FZI Karlsruhe, Dept. ESM, Haid-u.Neu-Str. 10-14, D-76131 Karlsruhe, GERMANY, <http://www.fzi.de/esm>, E-mail: esm@fzi.de

2. THE APPLICATION AND THE BUILDING STONES

Within the project an application scenario for an MPEG-4 Demonstrator was specified to highlight the benefits and features of MPEG-4 - based on Object orientation and Interaction, a typical slogan was: "Objects on demand". The second aspect of the Demonstrator was to visualize the outcomings of the different components developed within the project, which was internally organized similar to the MPEG structure, e.g. focused on Video, Audio or System development.

2.1 Infotainment Application

As interaction with AV objects is considered as the most important aspect of MPEG-4, infotainment applications, containing a combination of entertainment and information are suited for demonstrating the new MPEG-4 features. The user of such systems can configure and play with a multimedia environment and simultaneously gets information about a specific subject. The Interactivity involves e.g. the request of additional objects and changing of content of existing scene nodes.

MPEG-4 provides an ideal framework for infotainment applications:

- The composition concepts, which will cover both 2D and 3D, will be the base for mixing all kind of data types within a consistent object handling and user interaction paradigm.
- MPEG's tradition is to achieve the highest possible quality with existing techniques, which is only adequate for the demanding infotainment applications.

2.2 BIFS – the MPEG-4 scene and interaction specification format

BIFS stands for „**B**inary **F**ormat for **S**cene“ and offers a platform-independent description language for audio-visual 2D/3D scenes. BIFS represents a powerful extension of VRML 2.0, adapting most of its fundamental concepts and API elements. It enables to mix various MPEG-4 media together with 2D and 3D graphics, handles interactivity and deals with the dynamic changes of the scene in time. BIFS allows an object based transmission and interaction with synthetic and/or natural audio-visual objects.

There are several semantic main elements providing this functionality. Most important are nodes and routes:

Nodes represent the various audio-visual objects in a scene, which manages these nodes in a tree-structure. There are more than 70 different types of nodes, such as media nodes (e.g. `AudioSource`), geometrical nodes (e.g. `Rectangle`), interactive nodes (e.g. `TouchSensor`), executable nodes (e.g. `Conditional`) and structuring nodes (e.g. `Group2D`). All nodes have attribute fields which describe the properties of each instantiated node object. The `Circle` node, for example, has a `radius` field to set the size of each instantiated circle. The following example describes a simple scene containing a red `Circle`.

Routes mainly implement the interactivity of a scene. They are used to define the event-handling serving as routes between sender and recipient of an event. A route passes a sender-node's `eventOut` value to the specified recipient-node's `eventIn` value. The following (abstract) example should help to understand the BIFS event-handling:

```
DEF mySwitch TouchSensor {}
DEF myLamp DirectionalLight{}
ROUTE mySwitch.isActive TO myLamp.on
```

Whereas VRML is interpreted as an scripting language, the textual BIFS code - for compression reasons - is binary encoded (using the BIFS encoder) first, the encoding result finally serves as input for the MPEG player. The scene's tree structure is not static, it can be varied by update commands which allow to insert, replace and delete nodes and to change field values of the instantiated nodes. BIFS has different nodes for 2D and 3D Objects, which simplify the node-handling and reduce the data size in 2D scenes.

BIFS allows object based transmission and interaction with synthetic and/or natural audio- visual objects. It is exactly determined by the semantic elements nodes and fields how the user can interact with the objects in the scene. There are more than 70 different types of nodes, such as media nodes (e.g. Audio-Source), geometrical nodes (e.g. Rectangle), interactive nodes (e.g. TouchSensor), executable nodes (e.g. Conditional) and structuring nodes (e.g. Group2D). All nodes have attribute fields which describe the properties of each instantiated node object. Routes mainly implement the interactivity of a scene. They are used to define the event-handling serving as routes between sender and recipient of an event. A route passes a sender-nodes eventOut value to the specified recipient-nodes eventIn value. For compression reasons the textual BIFS input format and the descriptions of the A/V-objects that contains e.g. the allocation of the respective media decoder type and buffer sizes etc. are translated to a binary format before processing the data by a BIFS decoder and the individual objects by a multiplexer that generates proper access units for the MPEG-4 decoders that are integrated in the reference software.

3. IMPLEMENTATION

Two different approaches for system integration were taken during the project term, the first being a flexible combination of a Java System framework with native Decoder-software, the second and final one a more MPEG-4 phase 1 related C/C++ implementation, which is oriented on first Standard drafts and the official IM1 Reference Software, but with unique extensions and adaptations related to the usage of dedicated European multimedia accelerator hardware and other project specific components.

3.1 Data Flow in general

Figure 1 shows the flow of information and the treatment of time of the MPEG-4 player in its version dated spring 1998. As a simplification not all buffers are shown besides the essential Decoding Buffers (DBs) and Composition Buffers (CBs). The arriving data is demultiplexed and the first resulting Elementary Stream (ES) - a BIFS stream - is placed in the DB that is intended for this kind of information. The BIFS stream can be partitioned in Access Units (AUs) based on the instant of time when they become valid.

The BIFS decoder parses the binary BIFS description and passes the relevant part for composition such as the detailed information about the various nodes to the *Compositor*. The other part of the BIFS contains external update commands (BIFS events) to modify the scene, e.g. to insert, to delete or to replace a node, an indexed value in a multiple field or a route or to replace the whole scene, all these are in the following designated as external events and further on not distinguished from user events, for instance generated by mouse or keyboard etc.

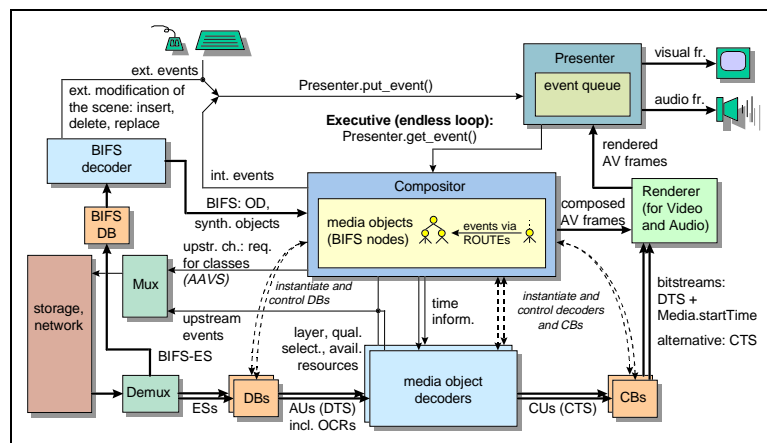


Fig. 1: Dataflow within an MPEG-4 Terminal

The composition relevant data within BIFS can also contain synthetic objects like 3D spheres including light effects etc. Based on this information, the *Compositor* knows which decoders are needed and instantiates them just as the associated Decoding and Composition Buffers (CBs). The sound and video bitstreams themselves can be stored in form of ES AUs in

the respective DB, which is the interface between the incoming multiplexed data and the decoding process. A decoder decodes at least one ES. Every AU has got an associated Decoding Time Stamp (DTS) provided by the encoder. After reaching the defined time, the AU is passed to the respective media object decoder.

The media object decoders generate Composition Units (CUs) with an associated Composition Time Stamp (CTS) attached to each one that is provided by the encoder. After that, the composed AudioVisual frames (AV frames) comprising the organized media objects including the integrated bitstreams are adopted and passed to the hardware by the *Renderer*, and finally presented to the frame application's screen resp. the speakers by sending the final scene reconstruction to the video display resp. the audio devices.

There is only a single event queue which is managed by the *Presenter*; external and internal events are put into this queue. The *Executive* class passes these events to the *Compositor*. Then the *Compositor* takes them into account with regard to the composed objects and their characteristics and processes the events chronologically.

Upstream events can be generated by either the *Compositor* or a media object decoder. They are processed by the *Multiplexer* and sent to the server. Upstream events can be messages containing information about the collision between media objects (`CollisionSensor`) as well as requests for classes, if the terminal is phase 2 / Java oriented.

3.2 IM1 familiarisation

For the integration and implementation of the application client, concepts were developed for adaptation to the IM1 core system with the project hardware and the integration of the project specific video and audio tools in mind. FZI intensified the maintaining and working with actual IM1 Reference software to base the new system on an IM1 core. This meant learning and exercising the VRML like scene description format BIFS and the internal composition structure. Basic exercises in BIFS to check the capabilities of the IM1 BIFS compositor for Interactivity implementation were a major part of the IM1 Implementation Software familiarisation. Using the official MPEG-4 Reference Software Implementation Group IM1 meant participation in the development and preparation of MPEG-4 demos in the framework of IM1, joint coding sessions during the MPEG meetings. The purpose was to show possibilities of BIFS composition as realised in den players/state of the art.

Concerning Demonstrator integration, the IM1 Core Classes and Communication Model were examined, the usage of MediaStreams and already for the ECAST Demonstration a prototypical IM1 based Demonstrator with EMPHASIS videodecoder integration was produced. Upto IBC Demo, improvements of the application / IM1-2D based demo system were done, rectangular and arbitrary shaped video playing as well as fade-in effect for visual scenes were introduced. Both tricky and extensive BIFS implementation as well as adaptation of the C code for demonstrating improved composition and interaction handling based on the IM1 compositor was realized.

Within MPEG available encoding software was installed and the necessary hardware setup to produce MPEG-4 compatible Audio/Videomaterial for the demos. We managed the overall content processing for the main demo, starting with Video encoding by an adaptation of the configuration files and parameters. The material was encoded by converting the CCIR YUV format video bitstreams to QCIF and CIF resolutions both in arbitrary and rectangular video formats. For an overview and test of the available material, all was encoded to 25/12.5 Hz framerate. Both raw and encoded sequences were donated to MPEG, and FZI offered additional encoding on request.

4. MPEG-4 INTERACTIVITY PROGRAMMING EXAMPLES

To produce a common framework for integration, a visible demonstration of the results of the different EMPHASIS work items and to highlight MPEG-4 functionality, the Demonstrator with following components was successfully integrated: Videosoftware, Audiosoftware, Systemsoftware, Contents. The underlying system is MPEG-4 Phase 1 IM1 oriented, to be as near to the standardisation activities as possible; the additional EMPHASIS TriMedia Hardware integration gives the system the necessary performance gain to enable realtime video decoding within the PC application system.

This chapter outlines the structure, composition and implementation of user interaction of the IM1 related MPEG-4 demo applications. The first part of this description is related to the BIFS scenes of the EMPHASIS main demonstration as well as

to a Teletext application that was already shown in an earlier MPEG meeting, but has been extended in order to demonstrate the IM1 2.5D player in a broadcast client-/server-scenario. The second part relates to a scene that provides the functionality to show video and audio individually as well as composed together, dependent from the user.

The system was demonstrated at ECMAST 98 in Berlin May 98 and in its final version within the exhibition at International Broadcasting Conference (IBC), Amsterdam, Sept. 1998. The application software containing the BIFS code and the A/V-content is available for MPEG members on systems reflectors support ftp site ftp.fzi.de (MPEG login).

4.1 Example scene: Main Square: Interaction and Composition

In the Main Square various objects are shown (fig. 2). Some of them are foreseen to provide the entrance to other scenes. A Roman and a knight start both telling their stories. The text is displayed by using vertically scrolling lines in a box. The user has e.g. the possibility to select the information icon to get more information about the objects. The launch of the scene is enhanced by a fade in effect. 18 filled horizontal rectangles are shifted from within the scene.



Figure 2: Main Square Objects

Several TouchSensors are placed one upon the other at the same place of the projection plane. Their drawingOrder is changed in a way so that the desired TouchSensor is always on the top. This is achieved by sequencing TouchSensors, which are defined within Transform nodes, in one Switch2D node. The Conditional nodes are activated by TouchSensors and thus the parameter of the Switch2D-field's whichChoice can be changed. Every single parameter of the whichChoice field stands for a Transform2D node that contains a TouchSensor. Dependent of the whichChoice's parameter is one of the Transform2D nodes active, the others are deactivated, i.e. not visible. Always one TouchSensor must be activated corresponding to the desired functionality.

4.2 Example scene: Teletext type Applications

The Teletext-Interaction application focus, which is similar to videotext as known from TV, lies within the MPEG-4 system on text, different fonts/backgrounds/colors and time-depending animation. Teletext application has been extended in order to demonstrate the IM1 2.5D player in a broadcast client-/server-scenario. A Teletext preview consisting out of different sub-channels has been designed in QCIF size to have a more impressive preview than a single JPEG. The preview is intended to be included as a sub-channel of the IM1 Mosaic.



Figure 3: Flight information

Clicking the plane symbol causes the arrival's panel moving slowly out of the screen (fig. 3), so that the departure's panel is shown. Clicking the plane again causes the state before and the arrivals overlay the departures. The info button on the

bottom right leads out of a special panel, back to the initial position. In successive steps more and more flights are displayed on the list. The departure and arrival title bar is continuously changing its colour from red to white. The state "take off" is marked by a blinking text.

The additional flights are added by using BIFS update commands at a certain time:

```
AT 2500 {  
    REPLACE PlaneDeparture2Nr.string BY "AF5732"  
    ...  
}
```

Blinking text is done by ROUTEing a TimeSensor to a ColorInterpolator. The latter transfers colour values to the Material2D node of a Text node. Therefore the colour changes from the background colour (invisible text) to visible colour.

Shifting in and shifting out of the arrival panel is done by a Position2DInterpolator that changes the translation values of a Transform2D node. The interpolator receives the key values from a TimeSensor that is activated by a TouchSensor.

4.3 Example Scene: Videodrome: Interaction and Composition

The Videodrome (fig. 4) provides the possibility for the user to interact with different objects, i.e. to move resp. to enable and disable them. Objects can be synthetic ones like the menu bar or a ball or natural ones like e.g. representative JPEG images for the video bitstreams or the latter itself as well as audio objects connected to visual objects.



Fig. 4: Videodrome

For the movement of the musicians the PlaneSensor could not be used because if it is activated the corresponding object is moved to the top of the hierarchy. Thus, it would be impossible to move an object behind another one. This is solved by subordinating two identical musicians in a Switch2D node (Transform2D -> Image2D -> url xxx).

Two Switch2D nodes are subordinated within a superior node. One of the Switch2D nodes contains the Transform2D nodes with respectively the same high drawingOrder (higher than the middle statue's one) and the other Switch2D the Transform2Ds with the lower drawingOrder. As a result of this it is possible to switch between the musicians with different drawingOrders to change their relative position to the statue and the other objects in the scene. The switching itself is done by using a TouchSensor attached to a narrow horizontal rectangle that extends to the whole width of the player's output window. If an object is being dragged over this TouchSensor a Conditional is activated that modifies the whichChoice parameter. The consequence is that one of the objects resp. the one with the desired drawingOrder is activated.

The reason for putting two identical objects within the Switch2D-node is that one contains a TouchSensor for changing the whichChoice field from zero to one via a Conditional. Thus, a Transform2D with the second object is activated. The latter is moveable because the TouchSensor is routed in this manner, so that via hitPoint_changed the translation of the main Transform2D on the top of the hierarchy can be modified. This main Transform2D is necessary since all objects shall have the same drawingOrder, even if they are not active.

Another kind of possible user interaction is to drag the musicians, singers etc. from the menu bar on the bottom part of the screen out to the garden and also to drag them back to the bar in order to remove the objects again from the scene. A further object is inserted in the main Switch2D node that contains also the other objects but it is displayed by a small JPEG button that represents the object. The Transform2D that represents the "small" object contains a PlaneSensor that passes the mouse movements directly to the translation field of the Transform2D. The "small" object is contained in a further superior Transform2D representing an invisible synthetic rectangle. This rectangle is arranged below the musician JPEG, is larger than the latter and equipped with an TouchSensor that is activated if the mouse crosses the border of the bar. As a result an isOver event is generated that changes the whichChoice parameter of the main Switch2D and the "big" musician JPEG appears on the screen. If the musician is moved to the direction of the menu bar and crosses an invisible TouchSensor that

extends to the width of the bar, it is sorted into the bar. The working principle of the upcoming menu bar if the mouse pointer is on the bottom of the screen is the same like in the AV Studio that is described below.

By pressing the menu bar's question mark button a help text on a half transparent background appears describing the principle possibilities of this scene to the user.

4.4 Example scene: Audio/Video Teststudio

Analogous to the Main Square a fade-in effect is used to uncover the scene. The grey areas as shown in the left part of fig. 5 consist of superimposed rectangles whose dynamic coordinates have been computed by using a C program. The middle part of figure 5 shows the initial screen of the Audio/Video Test Studio in the framework of the IM1 2.5D player. In the right segment the logo is replaced by one of the EMPHASIS video bitstreams.



Figure 5: Audio/Video Teststudio application

The animation of the speaker symbol in the upper left corner is realised in the following way: The right part of the speaker's JPEG is overlapped by a rectangle that has the same colour as the background. The rectangle's transparency is continuously changing by coupling a TimeSensor to a ScalarInterpolator. Immediately after loading the audio and video bitstreams (startTime 0) a TimeSensor is activated automatically. It is running until the end of the entire application (stopTime -1) and is repeating one time per second (cycleTime 1, loop TRUE). Because of a ROUTE it sends continuously fraction_changed events to a ScalarInterpolator during this time. The latter reacts to special time values (key [0.0 0.1 0.2 ...]) and sends one associated scalar value (keyValue [0 0 0 1 ...]) by the help of another ROUTE to a transparency-field of a rectangle's Material2D node.

The bottom part of the window consists out of JPEGs that are included by using Image2D nodes. They serve as start and stop buttons for audio and video, and to each of them is attached a TouchSensor to process mouse events.

If an audio sequence is started by clicking on one of the speaker symbols, a touchTime event of an associated TouchSensor is ROUTED to a startTime field of an AudioSource node and activates the audio. It can be stopped by the user in an analogous way by clicking on the stop symbol of the respective icon of the audio object.

The video sequences are shown in the image area in the middle. If no video is running, a JPEG logo is loaded. All playable media sequences are comprised and numbered consecutively as elements of the choice[] field of a Switch2D node starting with the index 0. The default is determined by the whichChoice field that is 0.

A mouse click on one of the video start buttons is triggering an isActive event of a TouchSensor that is ROUTED to an activate field of a Conditional node. The BIFS update command that is given in its buffer field is executed, a new value for whichChoice is set and thus the displayed video is exchanged. At the same time all the other video TouchSensors are deactivated, so that no other Video can be started. This happens in the same way like the activation. The REPLACE command within the buffer{} field sets the enabled fields of the TouchSensors to false.

All the H.263 video and G.723 audio files are multiplexed separately, i.e. a separate scenario file (scr-file) is needed, so that every object is allocated an own objectDescriptorID. By the help of the URL mechanism the respective media file is loaded at run time.



Fig. 6: Arbitrary shaped video bitstream

Instead of using rectangular shaped H.263 only it is also possible with an extended version of the IM1 2.5D player, as implemented in the recent versions, to display arbitrary shaped video bitstreams.

4.4 Lessons learned

A huge part of the realization of the MPEG system had to be devoted to the BIFS application and interaction programming. This however could be accomplished only after intensive study of the MPEG-4 reference software and its components. On purpose the Demonstrator was constrained to the possibilities of MPEG-4 phase 1, first to give the involved project partners the possibility to work within the official MPEG framework, and second to give a pure impression of the capabilities and features of the Standard in the current state.

The following table gives an overview of the implemented lines of code within the system. It goes without saying, that in times of object-orientation the metrics LOC has to be taken with care. However, there was no authoring tool for BIFS available - and also no debugger - at the start of implementing the BIFS scenes. The whole BIFS code was written and debugged manually.

BIFS		Video decoder		IM1/EMPHASIS player	
scenes	LOC	LOC		LOC	
<i>A/V Test St.</i>	4000	C code	header files	C/C++ code	header files
<i>Teletext</i>	12900	19900	5900	40400	20000
<i>Intro</i>	3100	among them			
<i>Main Square</i>	4300	<i>Multiplexer + BIFS encoder</i>			
<i>Videodrome</i>	7000			4400	1100
<i>Waterfall</i>	5100	<i>core</i>			
<i>Sea</i>	5100			7700	6300
total	41500				

Table 1: System size in LOC

5. THE TRIMEDIA INTEGRATION

The TriMedia is a multimedia coprocessor by Philips which was used within the EMPHASIS project in form of a PCI card within the Demonstrator PC. The communication between host PC and target hardware board was developed by testing Shared Memory/Callback functions and examination of the Scatter Gather Memory model.

A test application – a standalone TriMedia videc - was assembled from the available components, the TM optimized libraries and the PC version/framework. Following this, a TriMedia Videc DLL was integrated in the IM1 system core.

Work to improve the integration of the TriMedia videc to get a good performance in the complete system was supported by investigating the system setup (Videc, hardware, OS), which lead to the development of DMA-Transfer solution between PC and TM memory. Finally, concerning a moderate scene with video, a performance gain of about 30 % in contrast to usage of the general purpose CPU only was achieved and enabled real time video display on a standard PC environment with TriMedia.

The TriMedia optimized videodecoder consists of two libraries: the first contains the optimized modules of the decoder itself and the second is a collection of optimized modules of the renderer. In order to generate the complete version of the TriMedia optimized videc both libraries have to be linked within the TCS. The resulting application **videc.out** gets downloaded and started on the TriMedia board by the IM1 Player.

The figure below shows the data flow between IM1 and the TriMedia Videoc:

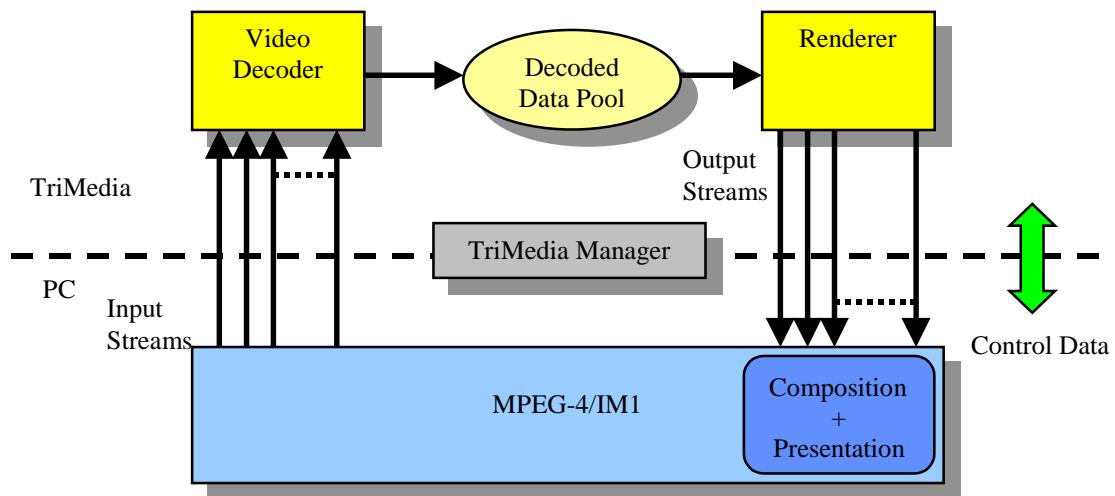


Fig. 7: IM1/ TriMedia Data Flow

The IM1-Player sends Access Units of a distinct Input Stream to the TriMedia board. The video decoder receives these Access Units, decodes them and stores them in a buffer structure, called Decoded Data Pool (Dpool). The renderer scans the Dpool and renders a complete frame from the material. The resulting frame is sent back to the IM1 Player as raw RGB data. IM1 receives the frames and composes them in the complete scene. It follows the presentation of the complete scenario on the screen.

All this needs a lot of data exchange and synchronization. One has to imagine the videoc on the TriMedia as an application on a remote server. Each application on the board works independently of the host PC.

6. FORMAL SYSTEM DESIGN AND COMPARISON

The traditional scope of work for our research group within FZI is in the domain of Formal System Design and CASE. Being involved in a variety projects with industry we realized the necessity to bridge the gap between design and realization of control systems like e.g. in the aerospace sector, and the new possibilities of software-based, distributed multimedia systems. For example, in the automotive area these application areas overlap in software development for modern car communication concepts, and we want to support the joint system design by common view and computer supported comparison of potential methods and standards as needed. In fact, within EMPHASIS some effort was dedicated to produce a system model with a classical CASE tool, capable of Simulation and Code generation.

The theoretical background will not be detailed here, however we will sketch the basic concept and the already available functionality which is of interest for multimedia system development also.

The basic requirements our system were two-fold:

- Support tool evaluations by a common questionnaire and data base
- Support selection of appropriate tools for given applications

As depicted in figure 8, the system consists of the following components:

A general criteria catalogue is the base for evaluation strategy and procedure, for the tool selection and for database realization. The catalogue is grouped hierarchically in three levels with categories, criteria groups and basic criteria. The main categories in the current state of development are Methods, Model representation, Usability, Simulation features, Codegeneration, Toolcoupling and Miscellaneous. With the introduction of application- and user-profiles the catalogue can

be applied and weighted to given scenarios. The valuation module includes concepts of statistics and fuzzy logic, and provides the methods for calculation of tool assessment related to given profiles.



Fig. 8: Concept of the Evaluation Environment

The GUI supports the creation and edition of catalogue criteria, tool assessment data and profiles. For platform independence and to take advantage of object-oriented programming, the tool is developed in Java, an example of the implemented tool GUI is shown below.



Figure 9: GUI example

The experiences and applications from the work within the multimedia domain as described above lead to the inclusion of lower CASE tools and Coding environments to the Evaluation system.

7. ACKNOWLEDGMENTS

The MPEG-4 related work was performed within the EU/ACTS project EMPHASIS AC105, we thank all concerned partners for providing their modules and support for the Demonstrator integration. FZI had the pleasure to participate in the most useful efforts of MPEG-4 reference software implementation group IM1, without their work no MPEG-4 systems standard would be possible.

EMPHASIS donated all created Audio/Video material as well as all BIFS Interaction and application program code to MPEG and the material is used for ongoing IM1 work. An intermediate demonstration based on the IM1 related new Demonstrator was shown at ECMAST Berlin in May 98, the final demo took place at IBC Amsterdam in Sep. 98 with all integrated components and the TriMedia.

References

1. EMPHASIS Project Homepage - <http://www.fzi.de/esm/projects/emphasis/welcome.html>
2. MPEG home page - <http://www.cselt.it/mpeg>