# Upgrading Multimedia Data Handling Services of a Database Management System by an Interaction Manager

*Heiko Thimm and Thomas C. Rakow*

GMD – Integrated Publication and Information Systems Institute (IPSI)
Dolivostr. 15, 6100 Darmstadt
Germany

e–mail: {thimm, rakow}@darmstadt.gmd.de

**Abstract**

A multimedia database management system (MM-DBMS) supports the development of multimedia applications and the handling of multimedia data at runtime. It explicitly handles continuous data such as video, audio and animation. The user of a MM-DBMS gets all the support he needs to store, manipulate, and present such data. Hence application development is not forced to deal with the implementation of basic multimedia concepts and modelling primitives. In this paper we especially focus on the integration of an Interaction Manager component (IAM) into a DBMS (i) to present continuous data and (ii) to control their presentation. Presentation control is possible interactively from a control panel, an interactive query or an application program. The connection of the IAM to the functionality of our data model language and the handling of continuous data presentations and presentation control events in cooperation with other system modules is described. In our opinion, the proposed IAM represents an important step towards the integration of multimedia data handling into the services of a DBMS to realize efficient support for multimedia data presentations.

## 1 Introduction and Motivation

Suppose that continuous data to be presented are stored in a database and that the DBMS lacks support to present continuous data. Application programmers are forced to include continuous data presentation functionality in each program. This is not an acceptable solution with respect to program design and programming efficiency. Since the presentation processing implemented in the application program is transparent to the DBMS it cannot provide efficient support. For example despite

of the interruption of a video presentation, the DBMS continues the retrieval of the data from secondary storage. The programmer has to implement a typical producer/consumer application that includes parallel hardware. The producer retrieves the continuous data whereas the consumer presents the data. Thus an efficient interprocess-communication mechanism is necessary. The communication protocol required has to specify the way continuous data streams from the DBMS to the application program are initiated and controlled (halted, resumed, terminated, manipulated with respect to direction and speed). Agreements on the quality of service parameters and on the applied coding and de/compression techniques have to be ''worked out'' in order to enable a correct interpretation by the receiving application program. Considering these requirements, we can deduce that the protocol will be a very complex one.

Contrary to the former assumption, let us now think of a DBMS that itself is handling continuous data presentations. Users just issue presentation requests while the DBMS itself is dealing with the datastreams, the decompression and interpretation of the data and intramedia as well as intermedia synchronization. Thus, functionality to present continuous data and control their presentation is not an implementation task. Efficiency of the datastream handling is an aspect of the DBMS implementation.

We will present a DBMS architecture which has a client-server architecture and features an IAM component. The IAM is contained in both, the client and the server. However, for the IAM there is no differentiation necessary since the Object Manager (OM) component makes the differences transparent to the other components. The IAM internally provides support to present continuous data and to control its presentation. Presentation control is possible interactively from a control panel, an interactive query, or an application program. Continuous media and input- and output devices are modelled in our DML. Processing requests can be issued to the IAM via DML objects or the query language.

If the *initiation of a new presentation* is requested, the IAM first checks the availability of the required output device. A message is provided to the user if the device is not available. If it is available the IAM generates a temporary object called *Presentation Object* that represents the initiated continuous data presentation. Information about the presentation become property values of the Presenta-

tion Object. A presentation control panel is provided by the IAM if demanded within the presentation initiation. Then the IAM instructs the OM to load the data into the buffer. Next the continuous presentation of the continuous data units (concerning *video/audio* such a unit is *a single frame/a certain bunch of audio samples*) is started and continuously driven by the IAM. The actual presentation of the units is performed by the involved output device. Thus the handling of the data formats (e.g. for data decompression) and the management of devices is transparent to the users. Due to its large size, continuous data is buffered in portions. Hence, after one portion has been presented the next one has to be loaded into the buffer. This is requested by the IAM which is closely interoperating with the OM. This cooperation is based on a certain protocol which makes the internal datastream handling transparent to the users.

The means of *controlling a continuous data presentation* are events which are issued to the corresponding temporary Presentation Object. By utilizing their own knowledge about the presentation, Presentation Objects filter the events that require a change of the IAM's processing from those that do not. Relevant events that have impact on the IAM's current processing cause presentation control requests. The Presentation Objects issue these presentation control requests to the IAM which adjusts its operation accordingly with respect to the involved presentation. Furthermore, it delivers information to the OM which adjusts its data buffering strategy accordingly as well. For example, concerning a stop-request, the presentation is interrupted and the next portion of the continuous data is not loaded any more. Or with respect to, e.g., a request claiming a video presentation with double presentation speed only every second frame is loaded and presented. The interrupt handling required for this is transparent to the users since it is hidden within the concept of an event. Built-in datatypes for continuous data of our data modelling language provide operations that are connected to the IAM's services. Thus at the data modelling level, the IAM makes the presentation of continuous data and its control transparent to the users. The current presentation state provided by the properties of the Presentation Object can be used within query specifications.

Currently in the AMOS-Project at GMD-IPSI, we are developing an object-oriented and distributed MM-DBMS [RM 93]. Already existing components of the object-oriented DBMS prototype VODAK [KNS 89, KNS 90, KN 90, MRKN 92] have to be modified. The IAM described in this paper

represents one of the additions required for the integration of multimedia data handling into the services of the DBMS.

This paper is organized in the following way. In section two, related work is described and compared to our approach. In section three, we derive the main requirements for our IAM component by looking at specific properties of multimedia data and multimedia computing. In section four, the architecture of the entire VODAK MM-DBMS is introduced. Section five covers VODAK's data model language support for the handling of multimedia data. In addition we illustrate how the IAM's services can be utilized via this language. We also show the data and control flow within the system for performing and controlling continuous data presentations. The IAM's operation and cooperation with relevant other system-internal components and its internal structure are described as well. Concluding remarks are given at the end where we also outline our future work and provide some references to multimedia projects in which we are involved.

## 2 Related Work

References in the literature related to our approach for an IAM are not very frequent. [CHT 86] suggests a Presentation Manager providing means for effective multimedia object presentation and browsing on the screen of a workstation. The Presentation Manager treats symmetrically objects which are mainly composed of text and objects which are mainly composed of voice. However, the proposed Presentation Manager, a component of the MINOS multimedia information system, represents a very specialized approach. Support for the continuous data is hard-wired in the system and presentation control is not considered. Furthermore, support for multimedia application programming utilizing the MINOS Presentation Manager has not been documented.

In [WKL 86, WK 87], the Multimedia Information Manager (MIM), a component of the ORION object-oriented DBMS, is introduced. The integration of the new datatypes is accomplished through a set of definitions of class hierarchies and a message passing protocol not only for the multimedia capture, storage, and presentation devices, but also for the captured and stored multimedia objects. This way, a high degree of flexibility is achieved since new storage or presentation devices are easily included by providing the corresponding types as subtypes of the existing types. This approach is

very promising, but it remains more or less a collection of classes. Specific database issues such as query processing, user interaction and architectural implications are not considered. Furthermore, it can be questioned whether the modelling of devices down to the level of methods such as *get-next-block* lead to efficient realizations. In the case of continuous media such as video or audio, this is just not feasible.

A presentation environment for hypermedia objects stored in a remote database and exchanged on a broadband network is proposed in [MPR 92]. The presentation control is handled via events like in our approach but no DBMS support is described. In [LG 90], a technique for formally specifying and modelling the temporal composition of multimedia data is proposed. Furthermore, a strategy is presented for constructing a database schema to facilitate data storage and retrieval of data elements based on the intermedia timing relationships established by the proposed modeling tool.

Several other publications that do not focus specifically on MM-DBMS are related to the presentation and control of continuous data. An object-oriented framework for composite multimedia is described in [Gi 91]. Composite multimedia is constructed from multimedia primitives and temporal transformations. Active objects based on real-time processes are proposed as multimedia primitives. [ATWGA 90] describes a resource model providing a uniform basis for reserving and scheduling resources in both networks and hosts towards support for continuous media in general-purpose distributed operating systems. [GC 91] provides a theoretical framework for the real time requirements of delay-sensitive multimedia data. [Ste 90] describes synchronization properties in multimedia systems. In [PE 91] a variable rate strategy for the retrieval of audio data is proposed which dynamically adjusts audio fidelity to match variations in available retrieval bandwidth. The design and implementation of a continuous media player for UNIX workstations is described in [RS 92]. The project DOM deals with distributed multimedia object management [MHB 91].

# 3 Requirements for the Interaction Manager Component

Specific properties of continuous data which have been analyzed in detail, for instance in [AK 92] and [TR 93], are (1) time-dependency and (2) high volume. Another aspect is that multimedia computing involves specific input- and output devices. Our requirements for the IAM that we give in

the following have been derived from these specific multimedia data properties and from specific aspects of multimedia computing.

**(a) Advanced Functionality:**

The IAM is primarily aiming at efficient support to capture, present and manipulate continuous data and to control their presentation interactively. Hence, for each continuous datatype, media-specific functions are required. Functionality for presentations and their control may not be limited to those functions known from consumer electronic devices like, e.g., presentation interruption, speed and direction manipulation. Concerning video, it should be possible to focus on a certain area, to zoom, to resize, ...etc. Since we have a client-server environment, it also should be allowed for users to specify individual quality of service parameters in order to match the available bandwidth. Functionalities for the manipulation of continuous data should allow to manipulate continuous data units, like for example fading out the loudness of an audio or pixel manipulations with respect to video frames. Moreover, copy and paste operations are desirable to tailor own videos and audios. Note that we will extend our current solution for the IAM with functionality to capture and manipulate continuous data later.

**(b) Intramedia Synchronization:**

The single units of continuous data like, for instance, the frames of a video-clip must not be presented at arbitrary points in time. Instead, the presentation has to conform to a certain rate. Concerning video presentations for instance, a minimum rate of 20 frames per second is required in order to achieve continuous movements. The IAM cannot perform continuous data presentations without being aware of the corresponding intramedia synchronization requirements.

**(c) Interruptability:**

Some of the presentation control commands are addressed to currently running presentations like the stop-command. Hence, interruptability is a central requirement for our IAM. Note that these commands depend on the multimedia datatype and that they can be time-critical. For example, concerning the stop-command, with respect to a running video presentation, the video should be halted exactly at the current position.

**(d) Efficient Internal Continuous Data Transportation:**

Despite the availability of efficient data compression techniques, handling continuous data still means dealing with high volume data. Hence the IAM's functionality for interacting with continuous data should be based on efficient internal data transportation. This concerns the transportation between storage hierarchies and the main memory, the transportation to/from output-/input-devices, and the network transportation. Thus, the number of internal copy-operations of continuous data should be minimized since they are extremely time consuming.

**(e) Management of Multimedia-Specific Devices:**

Many different physical devices are involved in interacting with multimedia data since one standard device cannot handle all kinds of multimedia data. Hence, the IAM has to deal with compression chips, equipment for analog or digital video/audio, presentation devices like loudspeakers, monitors, and windows, ... etc. While some devices can be used by several parallel presentations (e.g. video board, JPEG-chip), others cannot (e.g. speakers, microphone). The IAM has to allow and manage that one device can be shared by several parallel presentations. For devices that cannot handle parallel presentations, it has to consider device utilization conflicts that can occur (i) when an attempt is made to present several continuous data instances at the same time on one of these devices or (ii) when another (possibly foreign) application attempts to seize such a device which is currently busy.

**(f) Parallel Presentations:**

The exploitation of multiple media types in parallel is one factor of the attractiveness of multimedia applications. Hence in general, our IAM must support parallel presentations of continuous data. We have to differentiate between two kinds of parallel presentations. (i) parallel presentations without synchronization are e.g. several videos independently presented in parallel or an audio presentation that provides background sound for one or several video presentations. (ii) parallel presentations that have to be performed synchronously are, e.g., a video and its sound-track (see g).

**(g) Intermedia Synchronization:**

Continuous data presentations that should synchronously run in parallel require some synchronization information. The IAM must be able to handle this information. Hence, it needs to know the

means of the intermedia synchronization mechanism. Since the development of a concept for intermedia synchronization for VODAK has not been completed yet, this requirement has not been reflected in our IAM component yet.

# 4 Architecture of a MM-DBMS

A simplified graphical representation of the architecture of the VODAK MM-DBMS is given in figure 1 [TR 93]. The Object Manager module keeps the concrete location of the data transparent to the other system components. Thus, the presented architecture is adequate for the client configuration as well as for the server configuration. However, some parameters are different. For example, the time required for the data transportation.

Since VODAK is an object-oriented system, instead of performing operations on objects operations are executed by the objects themselves. This is achieved by sending specific messages to the objects. The Message and Event Handler Module is responsible for the realization of this mechanism. It determines the location of the receiver object, ensures that the receiver object and the method are

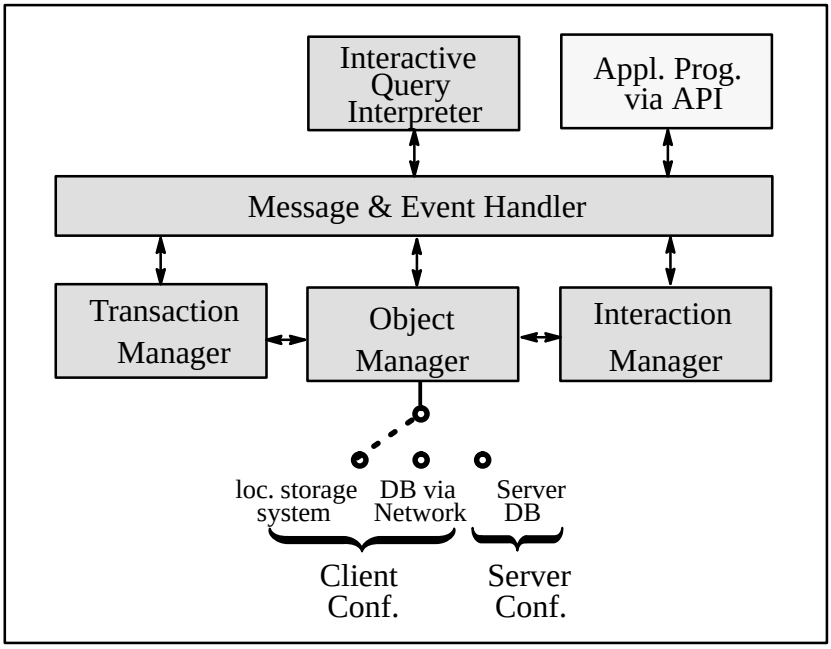


Fig. 1: Architecture of the VODAK MM-DBMS

loaded into the object buffer and that the method is executable. If so, the method is executed and the result is returned.

The Transaction Manager module provides an adequate transaction model for application programs and interactive queries accessing the data base. It allows the users to assume that any transaction, which is the execution of a program accessing the data base, is executed atomically – as if no other programs were executed concurrently – and reliably – as if there were no failures.

Data of the local filesystem (or any local storage device for which the local filesystem can be used as interface like a CD-ROM), a server-database or a data repository available via network utilization is made accessible for applications or queries by the Object Manager module.

The IAM module upgrades the multimedia data handling services of the DBMS. It offers support to present continuous data and control their presentation interactively.

The Interactive Query Interpreter module is handling interactive queries. Application programs that possibly contain queries can access the database via the application programming interface (API).

Note that certain changes and/or extensions of the shown modules' current implementation are required. The transaction management is extended to support access by application tools. Thus, check-out, check-in of large objects including persistent locks is supported. Furthermore, cooperative transactions have to be developed. A component for the management of continuous objects is developed and added to VODAK's Object Manager. These issues are subject to further research papers.

# 5 The Interaction Manager

First, we outline how our data model language supports the handling of multimedia data in general. Next, the exploitation of the IAM's services via this data model language is described. Then, concerning our entire system, we give an overview of the data and control flow caused by performing and controlling continuous data presentations. In the next part, we show how the IAM internally realizes the presentation of continuous data including presentation control. Within this context, the IAM's cooperation with other relevant system internal components is outlined as well. At the end of this section, we provide an overview of the internal structure of the IAM component.

## 5.1 Multimedia Data Modelling in VML

***Continuous Data Modelling:*** The DDL and DML facility of VODAK, called VML (VODAK Model Language) [Kl 92b], currently has been enriched by built-in datatypes for continuous data. These built-in datatypes can be specified as datatypes of VML properties. Thus, continuous data are handled as property values of VML objects.

In contrast to trivial built-in datatypes for continuous data such as BLOB (Binary Large Object) [Sh 90], we emphasize semantically rich built-in datatypes that support continuous access to units of continuous data (i.e. concerning *audio/video,* this is accessibility to a single *audio-sample/video-frame*). Like the well known datatype ''integer'' which inherently provides generic algebraic operations, our built-in datatypes provide generic facilities for the interaction with continuous data.

***Modelling of Devices:*** Input- and output-devices are modelled as VML-objects. Control panels that allow to control the capture or presentation of continuous data on such a device are modelled as VML objects as well. Thus, devices can be identified via their OID. Although handled as VML objects, the functionality provided by input- and output devices is not completely encapsulated in VML methods. For performance reasons, at least at the current time, this is not a good solution. Instead the functionality of the devices is utilized via their API.

## 5.2 Interaction Manager Utilization via VML

As a part of the definition of a VML object's structure, continuous data itself is modelled as a property (see figure 2). A VML built-in continuous datatype is specified as the property domain. As a part of the VML object's behavior modelled as VML methods, there is the user-defined method *present(...).* The implementation of this method utilizes the operation *initiate_Pres(...),* which is part of the built-in datatype's functionality.

From the modelling point of view we can draw the following analogy: In order to increase the current value of a property of type integer by an arbitrary integer value, we can write e.g. the following VML method specification ''*plus(number: INT)*''. The algebraic operation ''+'', which is inherently provided by the built-in datatype *integer,* is used within the implementation of this VML method. In

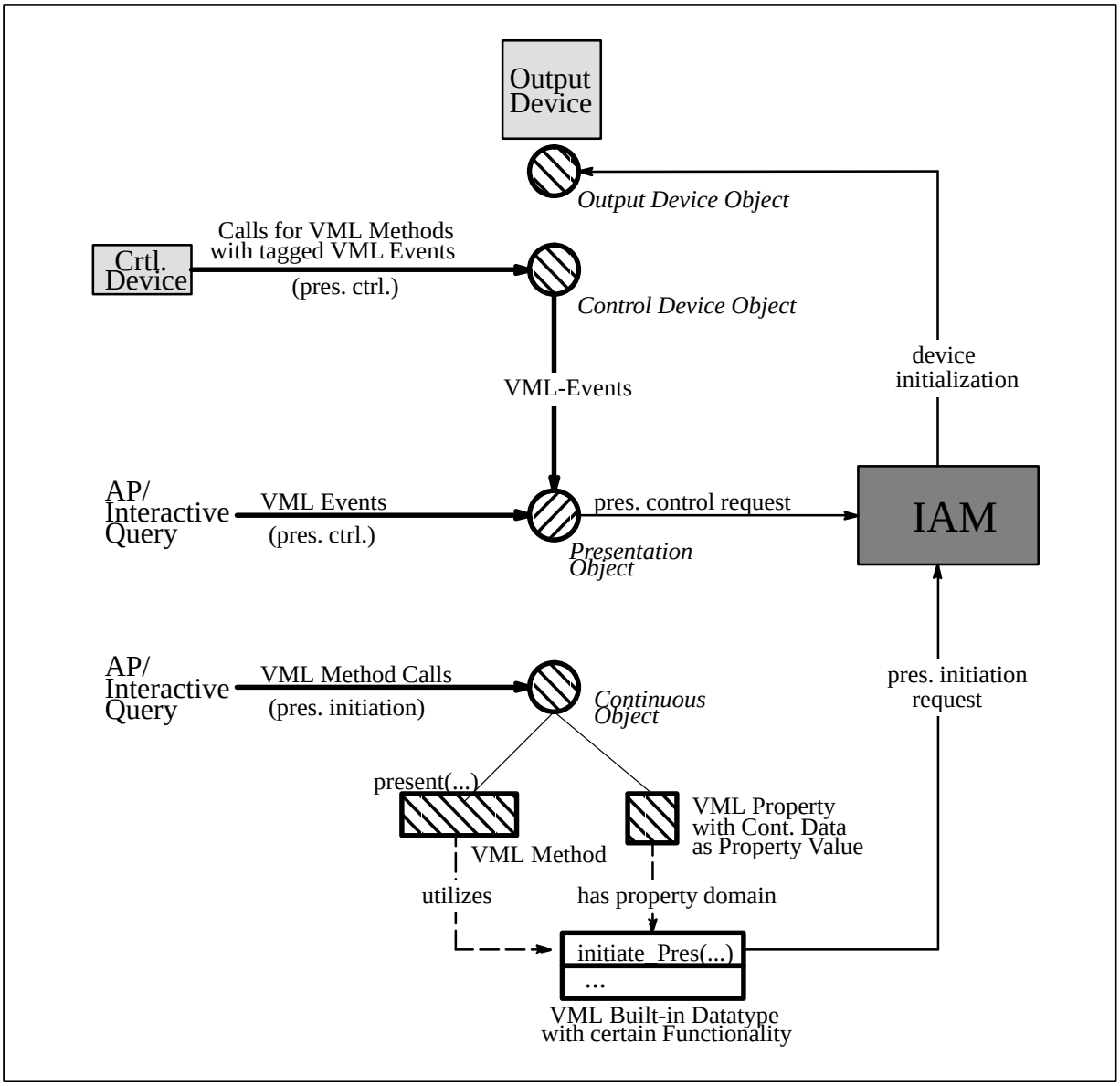


Fig. 2: Utilizing the IAM via VML

the following, we call the VML object defined in this way simply *Continuous Object* since continuous data is part of its structure.

The VML object that models the involved output device is called *Output Device Object.* At the level of our framework, it just provides an abstraction of the required device. In order to utilize a specific device, the OID of the corresponding Device Object has to be specified.

VML objects that provide an abstraction of a control device (e.g. video control panel) are called *Control Device Objects.* Control device facilities (e.g. buttons, sliders) are connected to VML methods of the corresponding Control Device Object. Thus, an activated control device facility causes a call for the corresponding VML method. VML events which are issued before the actual method is executed are tagged to these VML methods. The purpose of the method is the manipulation of the control device facility. For example, if the method is connected to a button, the method execution visualizes the change from a deactivated to an activated button.

For each continuous data presentation, there is a VML object which is called *Presentation Object.* These objects are dynamically generated at runtime by the IAM if a presentation initiation is requested. Moreover, they are temporary objects since they only exist as long as the corresponding presentation has not been killed. Presentation Objects provide the interface for controlling a continuous data presentation. Furthermore, they have relevant information about the corresponding continuous data presentation like, for example, the OIDs of the Output Device Object, Control Device Object and Continuous Object, the presentation state, direction, and speed, ... etc. Updating of these information is performed throughout the presentation session. The information is utilized in order to filter VML events that do not have any impact on the IAM's current processing from those that do. For example, if the last frame of a video presentation has been reached, a VML event requesting ''fast forward'' would be filtered out. However, if just half of the video is presented, the Presentation Object would issue the new presentation speed of the concerned presentation to the IAM. Hence, for relevant VML events, Presentation Objects issue corresponding presentation control requests to the IAM.

***Initiating Continuous Data Presentations:*** The user-defined VML method *present(...)* is called from an application program or an interactive query [Fi 92]. Since in the method implementation the built-in method *initiate_Pres(...)* is utilized, the method execution causes a presentation initiation request which is issued to the IAM. Certain parameters are passed to the IAM within this process like, for example, the OID of the desired *Output Device Object*, the OID of the desired *Control Device Object*, the quality of service parameters, presentation start position, speed, direction, ... etc. The availability of the required output device(s) is checked next. If this check is negative, the IAM

issues an error message. If it is positive, the device is initialized. Then the IAM generates a corresponding temporary *Presentation Object* and returns its OID. Relevant information about the presentation become property values of the Presentation Object. If an OID of a Control Object was given as parameter of the presentation initiation request, the IAM opens up the corresponding control device. Then the actual continuous data presentation is started and performed, as described in section 5.4 in detail.

***Controlling Continuous Data Presentations:*** Three possibilities for controlling a continuous data presentation are considered: (1) by an application program, (2) by an interactive query and (3) via utilization of a control device. In general, the means of requesting presentation control processing are predefined VML events. These VML events are issued to the corresponding Presentation Object. However, in (3) these VML events are not directly issued like in (1) and (2). They are issued as results of calls for VML-methods of the Control Device Object. Since they are tagged to these VML methods they are issued before the execution of the actual method starts. If a Presentation Object receives a VML event, it evaluates the consequences, i.e. it checks which presentation control request it has to issue to the IAM, if at all . The IAM handles the presentation control requests of its continuous data presentations. Depending on the requests, currently running presentations have to be changed, for example, with respect to speed or direction, interrupted, resumed, killed... etc. If a presentation is killed, the IAM automatically discards the corresponding Presentation Object.

Note for clarity, in figure 2 the Message and Event Handler Module has not been shown. However, it is utilized whenever a method call or VML event is sent to a VML object. Figure 2 shows the VML world as well as aspects of the physical world (output device, control device) and some system-internal aspects of the VODAK implementation. Bold edges denote the VML world.

## 5.3 Data and Control Flow

We believe that by our IAM component we can satisfy the requirements for interacting with continuous data for a large number of multimedia applications. However, we cannot exclude that sometimes the application programmer is forced or prefers to implement the functionality within the application program himself. For example, when the application programmer wants to utilize a special kind of

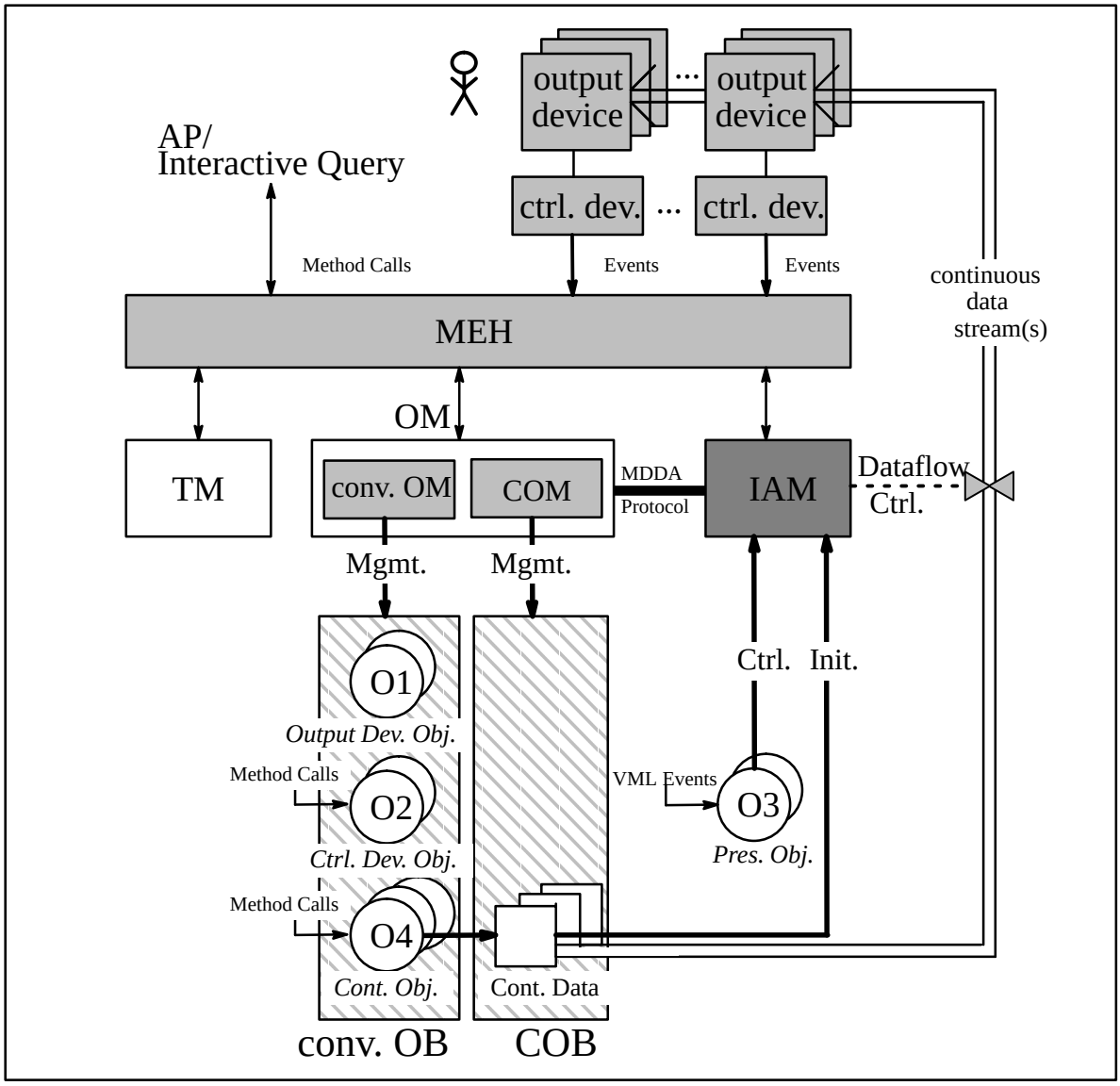


Fig. 3: Data and Control Flow

window not supported by the DBMS. Hence, we have to provide some flexibility for such applications, i.e. we have to offer some alternative way for the utilization of the continuous data. This has motivated our idea to support a direct access path to continuous data [MR 93].

We are able to provide direct access to continuous data by separately buffering continuous data in a special object buffer called Continuous Object Buffer (COB) which is managed by the Continuous Object Manager (COM) (see figure 3). The COM is a subcomponent of the Object Manager (OM). The Conventional Object Manager (conv. OM) which is handling discrete data is also a subcomponent of the OM. It is responsible for the conventional Object Buffer (conv. OB).

The IAM uses a predefined protocol called MDDA-protocol (Multimedia Data Direct Access) to interact with the COM. Together they satisfy the interaction requests issued within an application program or interactively via a control device at run-time. By applying the functionality of the involved output-device, the IAM controls the continuous data stream. The COB is the source and the output device is the sink of this datastream.

Alternatively, for application programs it is allowed to directly access the COB's contents by interacting with the COM via the MDDA protocol. In this case the IAM component simply is bypassed.

## 5.4 Performing and Controlling Continuous Data Presentations

Our explanation of the IAM component of the VODAK MM-DBMS has not yet covered the main aspects of the direct access to multimedia data. For simplicity, we always have assumed that the data to be presented constantly is available within the COB. However, this is not correct since due to the large size of continuous data, the COM normally can only load portions of continuous data into the COB. In the following, we will call such a portion of continuous data a sequence of (consecutive) presentation units (SPU). A single unit is, with respect to *audio/video*, a single *audio sample/video frame*.

An important aspect of the IAM/COM-cooperation based on the MDDA protocol is that both components work in parallel. Thus, the COM can preload SPUs.

It is not our intention to introduce the complete MDDA-protocol here (see [MR 93]). Instead, we show how it is used by the IAM within continuous data presentations and within presentation control processing.

The message flow diagram provided in figure 4 shows how a continuous data presentation, utilizing the IAM, is initiated and kept running.

In the following we briefly explain each processing step:

(1) There is a VML-object that has continuous data as one of its property values. Furthermore there is a user-defined VML method that initiates its presentation. This method is called. As explained in 5.2 this causes a corresponding call for the IAM.

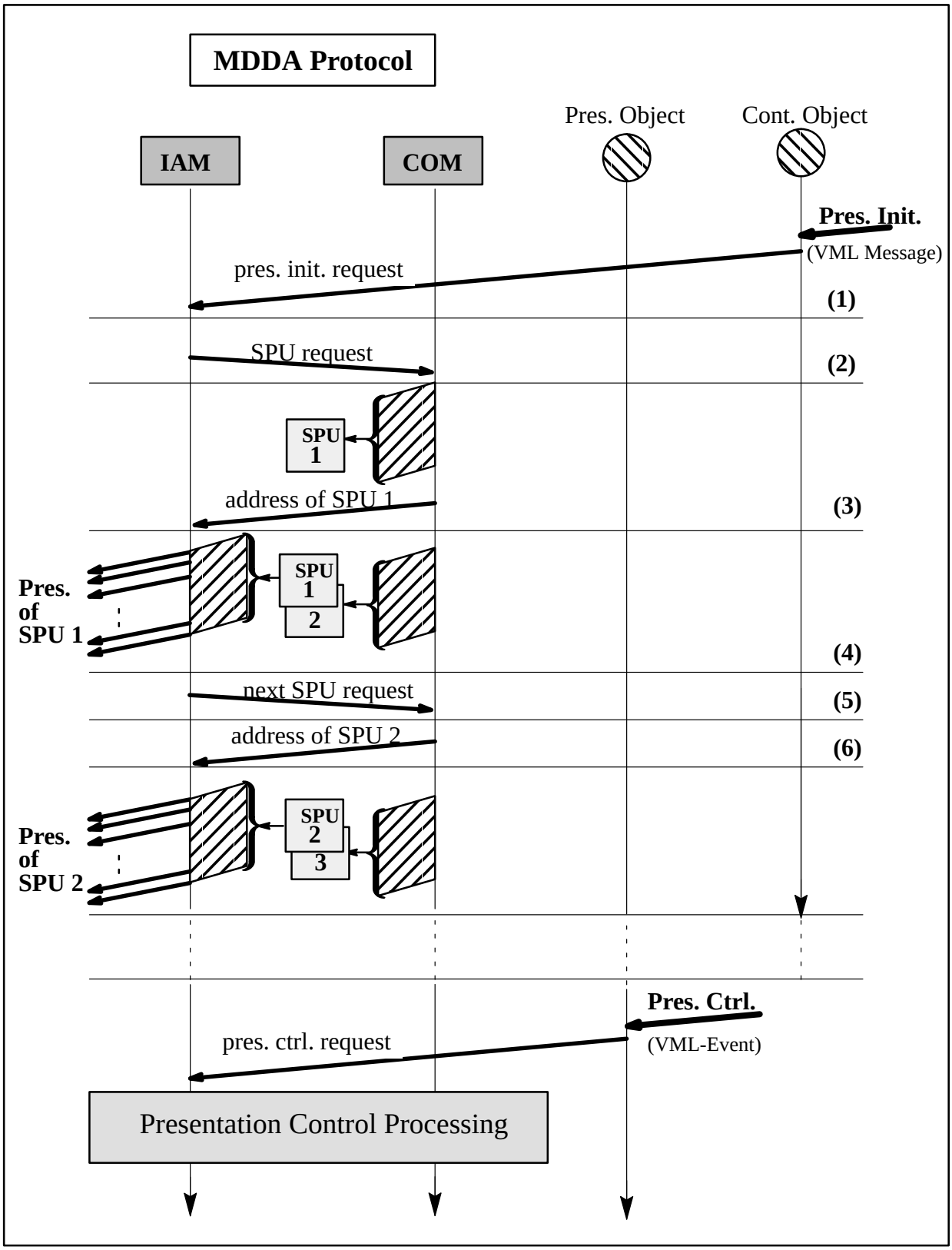


Fig. 4: Performing Continuous Data Presentations and Processing Presentation Control Events

(2) Assuming that the availability-check for the required output-device was positive, the IAM instructs the COM to load the first SPU into the COB.

(3) The loading procedure initiated in (2) is completed. The COM delivers a memory address to the IAM. Starting at this address, the pointers of the SPU's units can be consecutively found.

(4) The IAM presents the SPU by repeatedly calling, for each unit of presentation data, the presentation function of the output device. Note that this function, for performance reasons, is not encapsulated within a VML method. Its execution causes the presentation of the current unit of presentation data (concerning audio, the function can handle several audio-samples). At the same time, the COM is preloading the next SPU into the COB.

(5) The last unit of presentation data of the current SPU is presented. The IAM issues a request for the memory address of the pointer field of the next SPU's units of presentation data.

(6) The pointer requested in (5) is delivered. As long as no presentation control command is issued and the end of the continuous data has not been reached, the processing is continued at (4).

While executing this algorithm, presentation control requests can occur as it is indicated at the bottom of the message flow diagram. If so, the Presentation Object of the concerned presentation receives a VML event. If the VML event requires a change of the IAM's current processing, a presentation control request is issued to the IAM by the Presentation Object. Each control request received by the IAM has its individual processing which can cause the interruption, restart, or termination of the algorithm. Furthermore it can also cause the alteration of presentation parameters like speed or direction.

## 5.5 Internal Structure

A central requirement for the design of our IAM is interruptability in order to allow for interaction commands to control running continuous data presentations. Hence, continuous data presentations are handled in separate processes (figure 5).

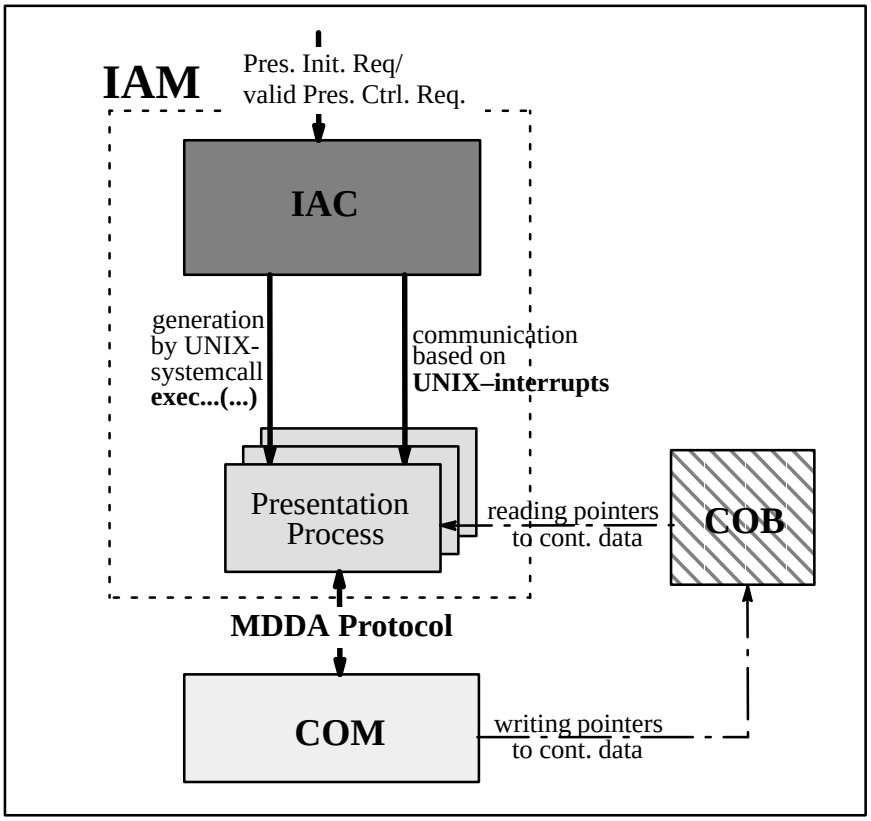


Fig. 5: The IAM's Internal Structure

All requests to be processed by the IAM are received by the Interaction Controller component (IAC). For each new continuous data presentation to be initiated, the IAC creates a separate presentation process running in parallel by utilizing the UNIX-systemcall ''exec...(...)''. Different types of presentation processes can be created which are specialized in fulfilling the specific needs of the presentation of an audio, a video, a sound-video or an animation.

Based on the predefined MDDA protocol, each presentation process is interoperating with the COM. The COM guarantees that, at any time, the continuous data to be presented is available within the COB. While the COM is writing the pointers to *audio-samples/video frames* into the COB, the presentation process is reading these pointers and presenting the *audio-samples/video frames.*

Presentation control requests which are issued from Presentation Objects are transformed into UNIX-interrupts which are issued to the corresponding presentation processes. Hence, interrupt handler routines are implemented in the presentation processes.

# 6 Conclusion

We discussed first why we think that an IAM in our sense is an important contribution towards the integration of multimedia data handling into the services of a MM-DBMS. Then by considering the specific properties of continuous data and specific aspects of multimedia computing we derived the main requirements of our IAM component. Next we introduced the entire system architecture of the VODAK MM-DBMS showing the integration of the IAM component. Then we gave a comprehensive description of the latter. First we covered VODAK's support for modelling multimedia data handling via its data model language and how the IAM's services can be exploited via this language. The data and control flow caused by performing and controlling continuous data presentations was explained afterwards followed by a description of the IAM's operation and cooperation with other relevant system internal components. At last we gave an overview of the IAM's internal structure.

The work presented in this paper is the first step of our research. As the next step we will extend the IAM's functionality by the capture and manipulation of continuous data. Since the capture of continuous data is inverse to the presentation of continuous data the same requirements with respect to time-dependency, the transportation and storage of continuous data have to be satisfied. Furthermore, we will integrate a mechanism for intermedia synchronization in order to support audio-visual presentations which are performed according to certain synchronization data.

Finally, let us point out that concerning all efforts towards our overall goal, namely the development of a MM-DBMS, our approaches or decisions are no armchair decisions. Instead, we always carefully consider the real needs of multimedia applications since we are involved in two big interdisciplinary multimedia projects called GAMMA (Globally Accessible Multi Media Archives) and MUSE (Multimedia Technology for System Engineering) [MUSE 93]. They allow us to constantly gain more insights about multimedia applications and serve as testbeds for our developments. So far it turned out that this is an excellent framework for this research to successfully make the VODAK object-oriented DBMS a true MM-DBMS.

# References

[AK 92] Aberer, K., Klas, W.: The Impact of Multimedia Data on Database Management Systems, ICSI, TR-92-065, Berkeley, CA, Sept. 1992

[ATWGA 90] Anderson, D.P., Tzou, S.Y., Wahbe, R., Govindan, R., Andrews, M.: Support for Continuous Media in the Dash System, Proc. of the 10th Int. Conf. on Distributed Computing Systems, Paris, May 1990

[CHT 86] Christodoulakis, S., Ho, F., Theodoridou, M.: The Multimedia Object Presentation Manager of MINOS: A Symmetric Approach, Proc. Int. Conf. on Management of Data, Washington, 1986, pp. 295–310

[DL 91] Dürr, M., Lang, S.M.: A Data Schema Approach to Multi-Media Database System Architecture, Int. Conf. on Multimedia Information Systems '91, Singapore, McGraw–Hill Book Co., pp. 311–317

[Fi 92] Fischer, G.: Updates in Object-Oriented Database Systems by Method Calls Queries. Proc. of 3rd ERCIM Database Research Group Workshop, Pisa, Sept. 1992

[GC 91] Gemmel, J., Christodoulakis, S.: Principles Of Delay Sensitive Multi-Media Data Retrieval, Int. Conf. on Multimedia Information Systems '91, Singapore, McGraw-Hill Book Co., pp. 147–158

[Gi 91] Gibbs, S.: Composite Multimedia and Active Objects, Proc. of the ACM OOPSLA '91, pp. 97–112

[Kl 92a] Klas, W.: Tailoring an Object-Oriented Database System to Integrate External Multimedia Devices, Proc. of 1992 Workshop on Heterogeneous Databases & Semantic Interoperability, Boulder, CL, Feb. 10–12, 1992,

[Kl 92b] Klas, W., et al.: VML – The VODAK Model Language, Version 2.0, working paper, GMD-IPSI, 1992

[KN 90] Klas, W., Neuhold, E.J.: Designing Intelligent Hypertext Systems U sing an Open Object-Oriented Database Model, Arbeitspapiere der GMD 498, St. Augustin, Sept. 1990

[KNS 89] Klas, W., Neuhold, E.J., Schrefl, M.: Tailoring Object-Oriented Data Models through Metaclasses, Proc. of the Advanced Database Systems Symposium, Kyoto, Japan, 1989

[KNS 90] Klas, W., Neuhold, E.J., Schrefl, M.: Using an Object-Oriented Approach to Model Multimedia Data, Computer Communications, Special Issue on Multimedia Systems 13(4), 1990

[LG 90] Little, T.D.C., Ghafoor, A.: Multimedia Object Models for Synchronization and Databases, IEEE J. Selected Areas Communication 8, No. 3, 1990

[MHB 91] Manola F., Hornick M.F., Buchmann A.P.: Object Data Model Facilities for Multimedia Data Types. TM–0332–11–90–165, GTE Laboratories Incorporated, Waltham MA, 1991.

[MPR 92] Marchisio, P., Panicciari, P., Rodi, P.: A Hypermedia Object Model And Its Presentation Environment, 3rd Eurographics Workshop on object-oriented graphics, Chamery, Switzerland, October 1992

[MR 93] Moser, F., Rakow, T.: The Multimedia Data Direct Access Protocol, in progress

[MRKN 92]Muth P., Rakow T. C., Klas W., Neuhold E.J.: A Transaction Model for an Open Publication Environment. In: Ahmed K. Elmagarmid (Ed.): Database Transaction Models for Advanced Applications. Morgan Kaufmann Publishers, San Mateo, Ca., 1992.

[MUSE 93]Lux, G.: MUSE – Multimedia Technology for System Engineering, ERCIM workshop, Trondheim, Norway, May 1993

[MW 91] Meyer-Wegener, K.: Multimedia-Datenbanken, B.G. Teubner Stuttgart, 1991

[PE 91] Park, A., English, P.: A Variable Rate Strategy for Retrieving Audio Data from Secondary Storage, Int. Conf. on Multimedia Information Systems '91, Singapore, McGraw–Hill Book Co., pp. 135–146

[RM 93] Rakow, T.C., Muth, P.: The $V^3$ Video Server – Managing Analog and Digital Video Clips, Description of Demonstration, accepted for SIGMOD '93, Washington DC, May 1993

[RS 92] Rowe, L.A., Smith, B.C.: A Continuous Media Player, AIV Workshop 1992

[Sh 90] Shetler, T.: Birth of the BLOB, BYTE, Februar 1990, pp. 221–226

[Ste 90] Steinmetz, R.: Synchronization Properties in Multimedia Systems. IEEE Journal on Selected Areas in Communications Vol.8 No. 3 April 1990.

[TR 93] Turau, V., Rakow, T.C.: A Schema Partition for Multimedia Database Management Systems, GMD Report 729, St. Augustin, Feb. 1993

[WKL 86] Woelk, D., Kim, W., Luther, W.: An Object-Oriented Approach to Multimedia Databases, Proc. ACM, ACM, New York, May 1986, pp. 311–325

[WK 87] Woelk, D., Kim, W.: Multimedia Information Management in an Object-Oriented Database System. Proc. 13th Int. Conf. on VLDB, Los Altos, CA, 1987, pp. 319–329