# 10 DESIGN, IMPLEMENTATION, AND EVALUATION OF TOOMM: A TEMPORAL OBJECT-ORIENTED MULTIMEDIA DATA MODEL

Vera Goebel[†], Ilan Eini[‡], Ketil Lund[†], Thomas Plagemann[†]

[†]University of Oslo, UniK - Center for Technology at Kjeller
{goebel,ketillu,plageman}@unik.no

[‡]Avenir ASA, Oslo
ilan.eini@avenir.no

**Abstract:** Multimedia database systems (MMDBSs) must be able to handle efficiently time dependent and time independent data, and to support Quality-of-Service. Based on the requirements of the DEDICATION project, i.e., building a MMDBS for asynchronous distance education, we have designed the data model TOOMM (Temporal Object-Oriented MultiMedia data Model). TOOMM is a novel data model that integrates temporal concepts into an object-oriented multimedia data model. TOOMM supports three time dimensions: valid time, transaction time, and a new time dimension specifically tailored for multimedia data types called *play time*. In this paper, we describe the concepts, implementation, and evaluation of TOOMM for the distance education scenario of the University of Oslo.

## 10.1 INTRODUCTION

Distributed multimedia applications like News-on-Demand, digital libraries, and asynchronous interactive distance education will be an important part of the future information society. These applications must be able to handle efficiently complex, continuous, and time dependent data types like video and audio as well as time independent data types like integer and text. The management of the complex structures of multimedia objects is one of the most challenging research issues for multimedia database management systems

(MMDBMSs). Today, object-oriented database systems (OODBSs) are used to handle multimedia data types (MMDTs). However, current object-oriented data models are not able to model all the temporal and spatial aspects of complex multimedia objects [18, 21]. Both these aspects are essential for presenting multimedia objects. Therefore, several features such as synchronization mechanisms, temporal and spatial relationships between objects, decomposition and re-combination of objects, and corresponding specification languages need further research.

In order to overcome the lack of an appropriate data model, we have developed a temporal object-oriented multimedia data model called TOOMM (Temporal Object-Oriented MultiMedia data Model). In addition to existing temporal concepts of transaction and valid time, TOOMM supports the so-called *play time* dimension. The play time dimension places multimedia data elements, such as video frames or audio samples, into a temporal structure for multimedia presentations. Furthermore, the *logical data model*, i.e., classes (object types) and instances (objects) containing multimedia data, and the *presentation model*, i.e., specifying how multimedia data should be presented, are separated in TOOMM. This separation is in accordance with the basic data modeling concept of independence between the way the data is stored in the database, and how it is presented to the user. The advantage of this separation and explicit combination is that multiple specialized presentations based on the same multimedia data can be created without the need for replication.

The remainder of this paper is structured as follows: Section 10.2 describes the distance education scenario and related projects at UniK in order to illustrate the usage and benefits of TOOMM in MMDBSs. Section 10.3 presents background and related work. In Section 10.4, the concepts of TOOMM are presented. In Section 10.5, we summarize the most important implementation issues. Section 10.6 presents an evaluation of TOOMM for the distance education scenario presented in Section 10.2. Section 10.7 concludes this paper and gives an outlook on future work.

## 10.2 DISTANCE EDUCATION SCENARIO

### *10.2.1 Synchronous Distance Learning*

Distance education refers to all types of studies in which students are separated by space and/or time. The electronic classrooms [3] at the University of Oslo overcome separation in space by exchanging digital audio, video, and whiteboard information between two sites of the University of Oslo and one of the University of Bergen. Since 1993, the electronic classrooms are regularly used for teaching graduate level courses as well as for research on Quality-of-Service (QoS) support in distributed multimedia systems [17]. The main parts of each electronic classroom are:

- **Electronic whiteboard:** at each site there is at least one electronic whiteboard (100") that is used to display lecture notes and transparencies written in Hypertext Markup Language (HTML) format. Transparencies

consume about 200 KB (per transparency) and must be kept in the buffer while displayed at the whiteboard. When a section is displayed, the lecturer can write, draw, and erase comments on it by using a light-pen.

- **Document camera and scanner:** can be used from the whiteboard application to capture the contents of printed materials, e.g., a page of a book, and present it on the whiteboard.

- **Audio system:** microphones are mounted evenly distributed on the ceiling in order to capture the voice of all the participants. Audio is PCM encoded and is digitized using a 16 bits/16 MHz sampler, which results in a constant data stream of 32 KB/s.

- **Video system:** one camera focuses on the lecturer, and two further cameras focus on the students. A video switch selects the camera corresponding to the microphone with the loudest input signal. Two monitors are placed in the front, and two monitors are placed in the back of each classroom displaying the incoming and outgoing video information. A H.261 codec is currently used to digitize and (de-)compress video data.

Today, only synchronous teaching is supported, that means the lectures are transferred in real-time over an ATM-based network to the peer classroom(s) and vice versa. Consequently, all students have to be physically present in one of the classrooms during a lecture.

### *10.2.2 Asynchronous Distance Learning*

In the DEDICATION (Database Support for Distance Education) project at UniK, we extend the functionality of today's electronic classroom to support asynchronous teaching by using a MMDBS to store the lectures for graduate level courses. To allow maximum flexibility, all transparencies and scanned images that are used in the lecture, the interactions with the whiteboard in the classrooms, as well as video and audio streams from the different classrooms are separately stored in a MMDBS. The separate modeling and storing of different MMDTs enables independent retrieval of data, e.g., reading the transparencies of the first hour of a certain lecture. Furthermore, the entire lecture, i.e., all multimedia data types and according data elements, can be reproduced: audio and video streams from all classrooms are continuously retrieved and their presentation to the user is synchronized with retrieval and presentation of transparencies and interactions with the whiteboard.

In such a MMDBS, students are able to retrieve lectures at any time. They may search for interesting topics, and play back only parts of lectures. Depending on the student's end-system, network connections, and requirements of the students, different QoS specifications have to be supported. For example, one student might work at home and is connected via ISDN, i.e., 2 x 64 Kbit/s, to the server. The student has followed the lecture, has a hardcopy of the transparencies, and wants only to recapitulate the explanations of the teacher. Thus,

the student retrieves the audio stream of the particular lecture with maximum quality and the video stream with low quality, i.e., low frame rate. Another student might have missed the lecture and retrieves the full lecture, i.e., audio, video, whiteboard, and document camera, in maximum quality from a terminal that is connected via Fast Ethernet, i.e., 100 Mbit/s, to the server.

## 10.3 BACKGROUND AND RELATED WORK

In this Section, we shortly survey the most important concepts and approaches for temporal and multimedia DBSs which provide the basis and related work for TOOMM.

### *10.3.1 Temporal Data Models*

Conventional temporal data model concepts [14] such as temporal dimensions, temporal domains, multiple granularities, and temporal relationships are the temporal basis of MMDTs. TOOMM provides a formal temporal framework for MMDTs solving many problems related to time management of multimedia presentations. For temporal DBSs, there exist many suggestions to realize temporal capabilities in the data model for relational DBSs [22] and OODBSs [4, 8, 19]. Multimedia data models such as SGML/HyTime [15] and Mediadoc [12] are more concerned with MMDTs and do not have the precise semantics of time, like pure temporal data models. A number of scheduling and synchronization techniques have been introduced for authoring multimedia presentations such as interval-based, axes-based, control flow-based, and event-based models. TOOMM includes many parts of these models and combines their advantages.

Various models of time have been proposed, e.g., discrete, dense, or continuous time [11, 14]. The time values of time dimensions can either be discrete or continuous. The discrete time domain model assumes that each time value in the time domain is mapped to a natural number, and for any member of a discrete time domain, there is a unique successor and predecessor. MMDTs like video and audio are called time dependent data types since the abstractions of the real world they model are continuous. However, when we capture video or audio data, we only capture data for discrete instants of time.

In the real world, we assume generally that there is only one time dimension. In the context of temporal DBSs [6], two time dimensions are of general interest: ***valid*** and ***transaction time.*** Valid and transaction time are orthogonal, meaning they can coexist in a DBS (bitemporal DBS) without interfering with each other while increasing expressiveness. Valid time denotes the time a fact is true in reality. Transaction time is the time during which the fact is stored in the database.

Conventional DBSs support time values as date and time of the day. Dealing with time values of different granularities creates certain problems such as how to handle the semantics of operations with operands of different granularities, and how to convert one granularity into another. Another important issue is the choice of the master time unit. All other time value granularities must

be related to the master time unit either directly or transitively by mapping functions [9].

### *10.3.2 Multimedia Data Models*

Many of the earlier works on MMDBSs are done in the context of multimedia document systems or as multimedia extensions for existing DBSs [7, 18]. More recent works on multimedia modeling mostly concentrate on developing models for dynamic elements of multimedia data presentations associated with continuous multimedia data. Three of the most relevant related multimedia data models for TOOMM are AMOS [1, 23], SGML/HyTime [15], and LMDM [20]. These systems support conventional alphanumeric data types (e.g., integer and real), *time independent data types* (like images, graphics, and text), and *time dependent data types* where we distinguish between two categories based on temporal characteristics: (1) *streams*, i.e., arrays of data elements that are intended for sequential presentation (e.g., audio, speech, and video), and (2) *computer generated multimedia data* (CGM), i.e., sets of operation specifications that a computer can execute over a period of time (e.g., animations and music).

An important aspect in all MMDBSs which is not sufficiently solved today is synchronization. Synchronization refers to the temporal relationships within multimedia objects and between multimedia objects. A presentation object can contain several multimedia objects which must be synchronized when multiple MMDTs are presented in parallel. Synchronization must be examined on many different levels. There exist two types of synchronization: (1) *intra-object synchronization*, i.e., temporal relationships within a time dependent multimedia object, and (2) *inter-object synchronization*, i.e., synchronization between multimedia objects.

In temporal reasoning, relationships between time intervals have been identified [2]. Little and Ghafoor [13] expand these temporal relationships to include n-ary relationships, although they are only of the same type. However, temporal relationships between objects of different types are additionally needed.

In a complex multimedia presentation, the start and stop times of the playback interval of one object can depend on the start and stop times of another object. When dealing with user interactions, this is especially useful since it takes into account operations and interrupts with unpredictable duration. Start and stop times of objects can either be expressed as delays relative to the global start time of the presentations, or as delays relative to other presentation objects. If events like start of a video playback are timestamped with delays relative to the global presentation start time, then after an edit operation that shortens or extends the entire multimedia presentation, the timestamps following the edit will be incorrect. Using temporal relationships the actual playback time is calculated at run-time.

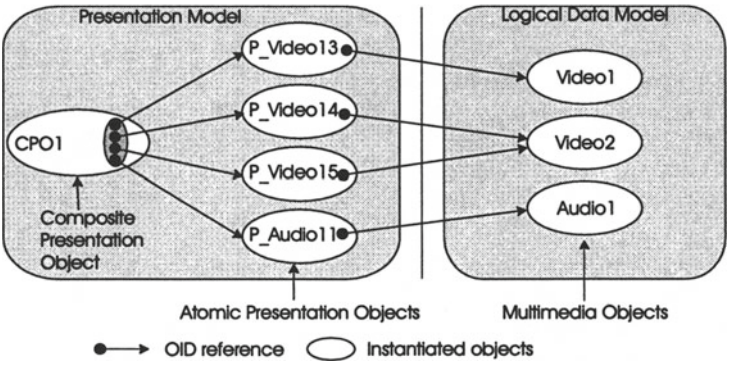


Figure 10.1: Relationships between logical data model and presentation model.

## 10.4 CONCEPTS OF TOOMM

TOOMM is a novel data model that integrates temporal concepts into an object-oriented multimedia data model. Objects in TOOMM comprise the properties of traditional object-oriented data models and three different time dimensions: valid time, transaction time, and play time. The play time dimension places *logical data units* (LDUs) of multimedia data, such as frames or audio samples, into a temporal structure for multimedia presentations. Furthermore, TOOMM is based on the following two principles:

- ***Separation of multimedia data from its presentation specification:*** This principle is in accordance with the basic data modeling concept of independence between the way data is stored in the database, and how it is presented to the user. Objects containing multimedia data are instances of object types from the logical data model. Objects instantiated from the presentation model specify how multimedia data should be presented. We differentiate between *atomic presentation objects* (APOs), that describe the presentation of single multimedia objects, and *composite presentation objects* (CPOs), that contain collections of presentation objects and metadata [5, 10], and support the correctly synchronized playback of multimedia data. Fig. 10.1 gives an example how the objects from the logical data model and presentation model are related. Summarizing, this principle supports multiple specialized presentations that are based on the same multimedia data which is only stored once in the MMDBS.

- ***Separation of multimedia data from its temporal information:*** Time dependent multimedia data include inherently temporal information, like a video that has a particular frame rate. In TOOMM, we detach the temporal information from the data, e.g., video frames, in or-

der to enable reuse of data in contexts with other timing constraints. For example, a video frame might also be used as an image in another multimedia presentation. Therefore, each video frame object - or generally each MMDT object - is associated with a timestamp via a *time associator* (TA) *object.* Fig. 10.2 illustrates the schema definition of a single video object.

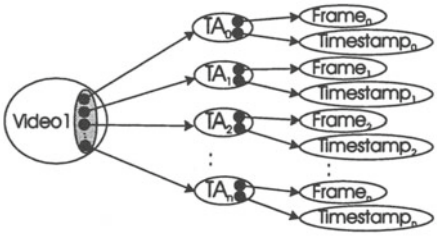


Figure 10.2: Modeling a single video object in TOOMM.

The following sections give a detailed description of the main elements of TOOMM, i.e., logical data model, play time, and presentation model.

### *10.4.1 Logical Data Model*

The logical data model consists of an extensible class hierarchy of the most common MMDTs such as video, audio, animation, music as well as basic abstract data types (ADTs). In this hierarchy, we differentiate between the following three main categories of MMDTs:

- *play time independent multimedia data types* (PTI_MMDTs),
- *play time dependent multimedia data types* (PTD_MMDTs), and
- *components of PTD_MMDTs,* which are for simplicity denoted later on as *components.*

ADTs that have static appearance during their presentation are said to belong to the PTI_MMDT category (see Fig. 10.3). Basic ADTs such as integer, real, boolean, character, and long also belong to the PTI_MMDT category. Additionally, TOOMM supports PTI_MMDTs with temporal characteristics: each PTI_MMDT can be extended with valid time and transaction time dimensions. Such an extension of time independent ADTs and PTI_MMDTs with temporal characteristics enables TOOMM to model data history and versions.

Unlike PTI_MMDTs, PTD_MMDTs comprise all types that have dynamic appearance during their presentation. In Fig. 10.3, the object types audio, video, music, and animation are shown as examples of PTD_MMDTs. Based on the PTD_MMDTs temporal characteristics, these object types are further

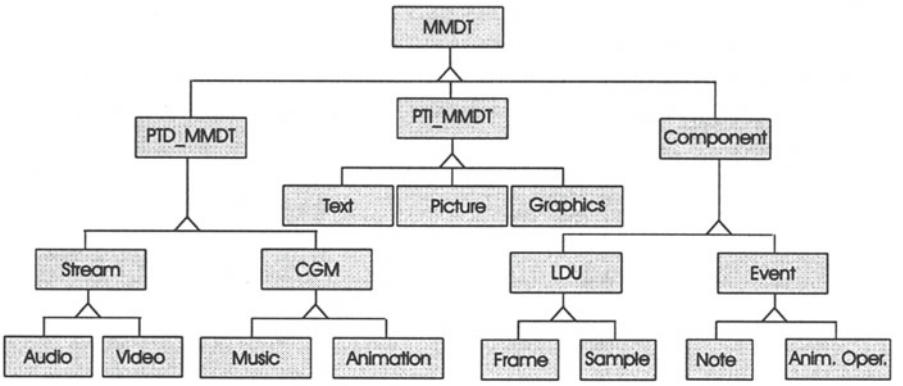


Figure 10.3: Logical data model type hierarchy.

classified into either *stream* or *CGM* sub-category. Stream objects are always related to a discrete time domain and are of periodic nature. Its components have to be presented at a constant rate and each single component has to be presented for a fixed duration. For example, all frames of a video object have to be presented with a rate of 30 frames per second (fps), and each frame has to be presented 1/30 seconds (s). In contrast, components of CGM objects are related to continuous time domains and can be presented in an arbitrary manner, i.e., they are a-periodic. The presentation duration of components is not fixed, and components can be presented sequentially or concurrently. For example, an animation of the space probe Voyager traveling through the solar system and adjusting its antennas might be composed out of three animation operations: one that generates the moving planets of the solar system, one that moves Voyager, and one for the adjustment of the antennas. The solar system and Voyager movements will be performed during the entire animation, but the antennas will only be adjusted from time to time for a short duration.

Component object types are classified according to the sub-category of the PTD_MMDTs they belong to. Component object types of stream object types are denoted LDU object types and component object types of CGM object types are denoted event object types (see Fig. 10.3). In other words, a stream object like video includes a set of references to LDUs and a CGM object like an animation includes a set of references to events. By placing components on the same level in the MMDT hierarchy as PTI_MMDTs and PTD_MMDTs, we achieve data independence and data reuse. All the object types in the PTD_MMDT category have corresponding object types in the Component category. This relationship exists for the PTD_MMDT object types introduced in this paper, because of their inherent temporal nature based on Components. Since the PTD_MMDT category in TOOMM can be extended with new ob-

ject types, the temporal nature of each new object type that is added must be investigated to decide whether it needs a new corresponding Component object type or not. For example, a video that has been originally stored with 30 fps can be easily (and without redundancy) provided with different frame rates, e.g., 10 fps, by creating a video object that contains only references to every third frame and adjusting the LDU_duration to 1/10 s. The provision of different frame rates in turn enables us to adjust the presentation and the amount of data to be retrieved to the user's QoS requirements.

The logical data model of TOOMM comprises in addition to this MMDT hierarchy two further important features: (1) metadata to describe the contents of multimedia data and (2) temporal information. PTI_MMDT objects and PTD_MMDT objects can contain references to metadata, i.e., text, that describes the contents of the multimedia object. For PTD_MMDT objects, two play time timestamps are used to relate the content description of a certain interval, e.g., a scene in a video, to the data of this interval. The following five types of temporal information are supported in TOOMM: (1) duration needed to present multimedia data; (2) duration needed to present the smallest LDU of multimedia data, which is not further decomposable; (3) data types may also contain additional time dimensions such as valid and transaction time; (4) object versions; (5) QoS information such as resolution, frame rate, and picture quality for video.

### *10.4.2 Play Time Dimension*

The valid and transaction time dimensions are not sufficient to model the temporal nature of multimedia objects and its diversity in time granularities. For example, the video standards NTSC and PAL work with different frame rates (NTSC 30 fps and PAL 25 fps) and time granularities of 1/30s and 1/25 s respectively. Audio is generally based on a much finer time granularity, e.g., the sampling rate of PCM coded audio is 8 kHz and CD-quality audio has a sampling rate of 44.1 kHz, which means time granularities of 1/8000 sec and 1/44100 sec. Therefore, we have introduced the play time dimension to handle the temporal nature of time dependent data and different time granularities in a media independent manner. Play time is used in the logical data model and in the presentation model; it can be seen as the glue between the two models. In the presentation model, play time is used as a means to map different time granularities of various multimedia objects to the global time granularity. In the logical data model, the play time dimension is used to define a temporal order between all components of multimedia objects. Based on this temporal order, we calculate the relative playback times of all components. Since streams and CGMs have different temporal characteristics, we look at the play time dimension for these MMDTs separately.

Streams contain a finite set of LDUs that have to be presented with a fixed rate and each for a fixed duration. Each stream object has to specify the default duration an LDU has to be presented (at normal rate), which is called *LDU_duration*. LDU_duration defines the time granularity of the multimedia

object, because it is the play time dimensions equivalent to a chronon [11]. For streams it defines the interval between the presentation of two consecutive LDUs and is inverse proportional to the LDU rate. All LDUs of a stream object are ordered according to the normal playback mode by associating each single LDU with a play time timestamp via a TA (see Fig. 10.2). The moment a user initiates the playback of a multimedia object, the play time values of the LDUs are bound to the actual time.

CGMs contain a finite set of events that might be presented in an arbitrary manner, sequentially or concurrently, and all with possibly different presentation durations. Thus, CGMs cannot be associated with duration specification for all its components (as it is done in streams). However, each CGM requires the specification of a chronon, because the theoretically continuous time domain of CGMs cannot be implemented in reality, and it is necessary to relate all its temporal specifications to one specific time scale. These temporal specifications are actually start and stop times of events and are related via a TA to each event. Fig. 10.4 compares the usage of play time in streams and CGMs.

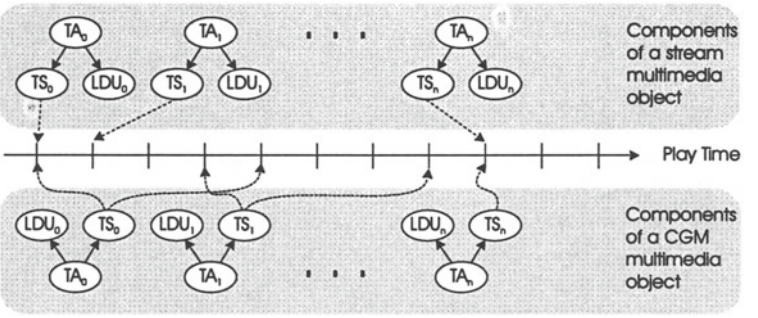


Figure 10.4: Usage of play time in streams and CGMs.

A major benefit of TOOMM is to combine streams and CGMs in a presentation in a uniform and flexible way. This is achieved through mechanisms for management of mixed granularities, and construction of CPOs for handling the presentation information of different MMDTs. The time scale used in the temporal specifications of a CGM is the play time dimension of that CGM. One granule in that time scale corresponds to a chronon.

### *10.4.3 Presentation Model*

The logical data model supports temporal information for single multimedia objects, like LDU_duration, because this (default) temporal information belongs inherently to the data. However, a single multimedia object might be presented in different ways, independent of its default temporal information.

For example, a video with 30 fps might be presented with the default rate or in slow-motion. Thus, we have to differentiate between the default temporal information of multimedia objects and temporal information that is used for a particular presentation of the object. Furthermore, temporal relationships between multimedia objects are not included in the logical data model to support unconstrained combination of multimedia objects in presentations. For instance, a particular video sequence might be presented with audio sequences (speech) and sub-titles in different languages. In order to promote data reuse in TOOMM, all temporal relationships that are relevant for presentations are part of the presentation model. Different presentations that use the same data can be independently stored, and the multimedia data is only stored once.

We differentiate between two object types in the presentation model: APO types and CPO types. APOs are the atomic building blocks of a complex multimedia presentation that is specified in a CPO. Each APO specifies the presentation of a part, or entire single multimedia object. Thus, for each multimedia object type, TOOMM provides one APO type. APOs typically contain information about:

- References to multimedia data in terms of play time using the global play time dimension that is defined in the corresponding CPO. APOs that refer to a PTD_MMDT object must specify a continuous sequence of LDUs that have to be presented via start and stop time. In this way, the APO can select a part of, or the entire data set of the multimedia object. APOs that refer to a PTI_MMDT object must specify the time when the PTI_MMDT object should be presented. For instance, a presentation object referring to a picture object should specify the time interval within the multimedia presentation the picture should be shown.

- QoS specification for multimedia data presentation. The QoS of the presentation can differ from the maximal quality of the stored multimedia data. The presentation model enables us to specify different (lower) QoS for a presentation. For example, a video that is captured and stored with 30 fps might be presented at 25 fps. Network QoS parameters such as throughput requirements, jitter threshold, skew tolerance, error tolerance, and synchronization requirements can be extracted from the data and metadata stored with TOOMM, and used to negotiate QoS requirements and guarantees by the presentation execution module. For example, presentation parameters such as frame rate or frame resolution can be reduced to satisfy limitations in the available resources for a specific presentation.

- Effects on the multimedia data, such as, fade in, fade out, or change in volume.

CPOs specify the structure of complex multimedia presentations. The main elements of a CPO are a set of APOs and a set of temporal relationships among these APOs. Additionally, it contains the definition of a master chronon,

termed *Master Time Unit Duration* (MTU_duration). The MTU_duration sets the granularity of the global play time which all the presentation objects must relate to. CPOs typically contain:

- Temporal relationships among multimedia data presentations. This is done through *temporal relationship objects* (TROs) which connect presentation objects, and specify their mutual temporal dependencies.
- Alternate multimedia data. For instance, if a video sequence has audio sequences in different languages available then the user should be able to choose between them.

Fig. 10.5 illustrates structure and usage of the presentation model. The extended entity-relationship diagram shows the type hierarchy of the presentation model and how the objects in the different parts of the data model relate to each other. The MMDT object type is the same as depicted in Fig. 10.3 but its sub-type hierarchy is omitted because of space limitations.

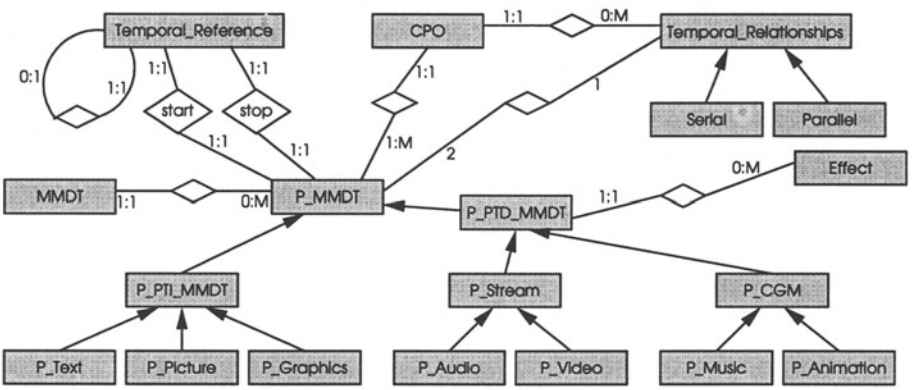


Figure 10.5: Extended entity-relationship diagram.

### *10.4.4 Comparison of TOOMM with Related Approaches*

TOOMM integrates and extends well known concepts from object-oriented, temporal, and multimedia data models to provide better DBMS support for multimedia applications. Especially, TOOMM utilizes concepts from T_Chimera [4] and TIGUKAT [16] temporal object models, and the News-on-Demand [15] multimedia object model. The T_Chimera object model provides the temporal (T) construct, which extends the type T with temporal capabilities. In TOOMM, we use this construct to add temporal capabilities to MMDTs. It

provides us with a concept to structure the temporal information of both data history and data presentation.

Presentation of multimedia data can be performed in many different ways. Hence it is beneficial to separate the logical data model of multimedia data from the presentation model similar to the view model of traditional DBSs. This is also done in the SGML/HyTime data model through their event schedule.

## 10.5 IMPLEMENTATION ISSUES

TOOMM is implemented on top of the OODBS ObjectStore. We have implemented a C++ class library that utilizes ObjectStore and provides DBMS support for common MMDTs. This implementation of TOOMM is based on the object definition language (ODL) of ObjectStore. The important advantage of using ODL is that the ODL definition can be used to produce the class implementations of the TOOMM object types in different programming languages by applying ODL preprocessors. The TOOMM object model is implemented in a client-server architecture where the server side is implemented in C++ (ObjectStore), and the client side can for instance be implemented as Java-applets that access the DBS over a network. In order to use TOOMM, an application developer must include the TOOMM C++ library files in the application programs, and an ObjectStore server must be available. In this section, we briefly review the most important implementation issues of TOOMM by illustrating object types that are necessary to present video.

### *10.5.1 Logical Data Model*

**10.5.1.1 Organizing Temporal Information.** In addition to the object types in the logical data model, we need the object type time values, timestamps, and the construct Temporal (T,M) to manage temporal issues (see Fig. 10.6). Time value object types are modeled like the T_discreteDeterminate Instant object type of the TIGUKAT temporal object model [8]. The Timestamp object type is abstract and never instantiated. Sub-types of type Timestamp can model time entities such as instants or intervals. Furthermore, they can contain as many time dimensions as desired.

```
interface Play_Time {
     attribute T_discreteDeterminanteInstant time;
};
interface Timestamp {
     ....
};
interface Temporal (T,M) {
     attribute Set<struct<T Timestamp, M MMDT>> m_history;
};
```

Figure 10.6: Time value, timestamp, and Temporal (T,M) interface specification.

In the Chimera Temporal Object Model [4], T_Chimera, data types are extended with temporal capabilities through the construct Temporal (T). In TOOMM, we extend the basic idea of the T_Chimera model with various temporal characteristics of MMDTs. The object type Temporal (T,M) is used to attach Timestamp objects of category T to MMDT objects of category M.

Object types in the Component category are normally associated with the LDU_Timestamp or the Event_Timestamp objects containing the Play_Time attributes. The structure in the Temporal (T,M) definition corresponds to the TA objects in Fig. 10.3.

**10.5.1.2 Type Hierarchy.** The type hierarchy of the logical data model comprises for video the following specifications (see Fig. 10.7): MMDT, PTD_MMDT, component, stream, LDU, LDU_Timestamp, and video. The type MMDT is an abstract super-type of all the MMDTs. When non-empty, the default_presentation attribute specifies how the content of the MMDT object should be presented if no other specification exists. The PTD_MMDT abstract object type is meant to be sub-classed by time dependent MMDTs such as audio, video, music, and animation. It contains general information for all MMDTs that need a temporal extent to be played out meaningfully.

The Component object type is sub-typed by all the PTD_MMDT components. The belongs_to attribute inherited by all object types in the Component category is used to reach the PTD_MMDT object that the Component object is a component of.

The Stream object type contains general information for data types having a constant LDU rate such as audio and video. The m_data (multimedia data) attribute is a set of pairs of LDU_Timestamps and LDU object types. In the m_data attribute the LDU instances are always instances of a sub-type of the object type LDU. For instance, the m_data attribute of the Video object type is realized by a set of (LDU_Timestamp, Frame) pairs.

LDU is an abstract super-class, which must be sub-classed by the specific Component type of the PTD_MMDT. Since streams usually are recordings of real events it can be meaningful to register their valid times and transaction times. The LDU_Timestamp must at least contain the play time attribute pt, which places its corresponding LDU on the play time dimension. In addition, the LDU_Timestamp object type can be defined with other time values. The user of TOOMM may create user-defined timestamps through sub-classing the Timestamp super-class.

The object type Video inherits the m_data attribute from the object type Stream. The LDU_duration attribute is inherited from the PTD_MMDT. These characteristics, globally described by the Video object type attributes, are shared by all frames belonging to the same Video object. The Frame object type is a component of the Video object type. It represents an image that during presentation of a Video instance must be displayed at a given time for a certain duration.

```
interface MMDT {
     attribute P_MMDT default_presentation;
};
interface PTD_MMDT:MMDT {
     attribute Temporal (CD_Timestamp, Text) content_description;
     /* A textual description of the content of multimedia data */
     attribute float LDU_duration;
     attribute integer duration;
};
interface Component:MMDT {
     attribute belongs_to;
     ....
};
interface Stream:PTD_MMDT {
     attribute Temporal<LDU_Timestamp, LDU> m_data; /* TA! */
     ....
};
interface LDU:Component {
     ....
};
interface LDU_Timestamp:Timestamp {
     attribute Play_Time pt;
     attribute Valid_Time vt;
     attribute Transaction_Time tt;
};
interface Video:Stream {
     attribute Compression_Scheme cs; /*Specifies how all the frames are compressed */
     attribute Coding c;              /*Specifies how the frames are coded */
     attribute Resolution res; /*Specifies horizontal and vertical resolution */
     attribute Color_Depth cd; /*Specifies how many colors the frames consist of */
};
interface Frame:LDU {
     attribute Bit_String Frame_data;
};
```

Figure 10.7: Interface specification of logical data model.

### *10.5.2 Presentation Model*

**10.5.2.1 Temporal Issues.** The definition of temporal references and temporal relationships is necessary to handle the temporal issues of the presentation model (see Fig. 10.8). Temporal reference objects refer to other temporal reference objects, but an invariant is that the references cannot be cyclic. In addition, the last temporal reference object in a list must point to the global play time dimension of a CPO object. It must be ensured that these invariants are maintained during the creation and updates of a Temporal_Reference object. If the value of reference is false, the object is a time point and the play time relative to the CPO presentation is found in the time_point attribute. Otherwise, the actual time point of the reference is as follows: deviation + ref.get_time_point(). The function get_time_point() is a recursive function that traverses a list of references and accumulates their deviation values. The function terminates as soon as it runs into the first Temporal_Reference instance that has the reference value set to false. The return value of get_time_point() can be expressed by the equation:

$$[reference = FALSE \implies get\_time\_point() = time\_point]$$
$$\wedge \quad [reference = TRUE \implies get\_time\_point() = deviation + ref.get\_time\_point()]$$

```
interface Temporal_Reference {
     attribute boolean reference;
     attribute Play_Time time_point;
     attribute Temporal_Reference ref;
     attribute CPO the_presentation;
     attribute long deviation; /*In units of the_presentation.MTU_duration */
     Play_Time get_time_point();
};
interface Temporal_Relationship {
     attribute P_MMDT m1;
     attribute P_MMDT m2;
     relationship CPO belongs_to inverse CPO::Temporal_Relationships;
     boolean integrity_test();
};
interface Sequential:Temporal_Relationship {
     attribute enum sequential_rel_type {before, after, meets, met_by};
     boolean integrity_test();
};
interface Parallel:Temporal_Relationship {
     attribute float skew_toleranse;
     attribute enum parallel_rel_type {equal, starts, started_by, finishes,
                                       finishes_by, overlaps, overlapped_by, during, contains};
     boolean integrity_test();
     ....
};
```

Figure 10.8: Interface specification of temporal issues.

Fig. 10.9 gives an example of how temporal references can be utilized. The resulting start times are: 10 for both P_Video1 and P_Audio2, and 15 for P_Text3.

We look now at the practical realization of temporal relationships from the data model. We create object types whose instances function as edges in an object graph where the nodes are multimedia objects. This approach supports only binary temporal relationships. The two P_MMDT attributes m1 and m2 are references to the two presentation objects whose temporal relationship we want to model. The belongs_to relationship relates the Temporal_Relationship to a CPO instance. The integrity_test() method must be reimplemented in the sub-types of the Temporal_Relationship object type. This test must basically check that the start and stop endpoints of the multimedia object referred to by m1 and m2 are according to the relationship type and that they belong to the same presentation.

The Sequential object type models all the serial temporal relationships [2]. The Sequential_rel_type attribute value must be set to the type of the serial temporal relationship that the instance of Sequential models.

The Parallel object type models all the temporal relationships where multimedia objects are played back in parallel. The parallel_rel_type attribute value must be set to the type of the parallel temporal relationship that the instance

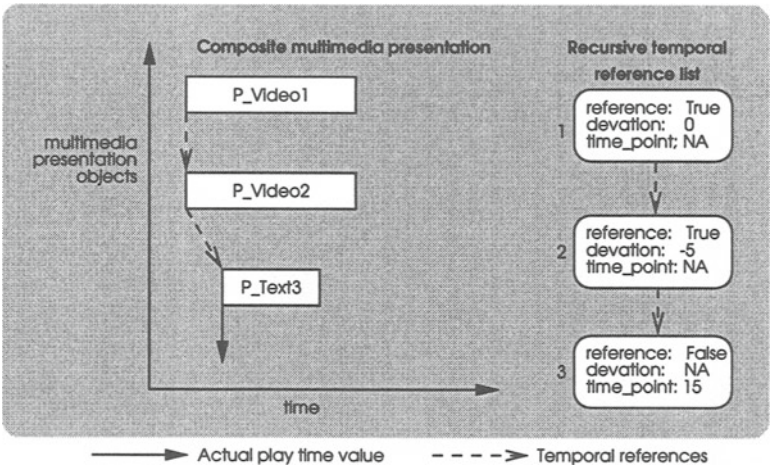


Figure 10.9: Example of temporal reference usage.

of Parallel models. The skew_tolerance attribute is an essential QoS parameter. The skew_tolerance unit is milliseconds. The skew between the presentation of two temporally related objects must be monitored at all times during presentation to avoid overstepping the skew tolerance limit.

**10.5.2.2 Presentation Multimedia Data Types.** Based on the previous specifications, we define the presentation multimedia data types that are necessary to present video objects: P_MMDT, P_PTD_MMDT, and P_Video (see Fig. 10.10). The presentation model (P_MMDT) consists of object types describing how the data should be displayed. The P_MMDT interface can be further sub-typed into object types used to present different MMDT data. An MMDT instance can be referred to by many P_MMDT instances. This accurately models the fact that a multimedia object can be presented in many ways.

The *presentation play time dependent MMDT* (P_PTD_MMDT) object type contains presentation information applicable to all play time dependent MMDTs. The start and stop attributes are relative to the start-time and stop-time of the MMDT instance referred to by mm_ref. If start equals zero and stop equals mm_ref.duration then the entire referred object is included. The speed attribute defines the speed of the presentation of the PTD_MMDT instance.

The Effects attribute is a set of (Effect_Timestamp, Effect) pairs. Some effects are global and can be executed by all play time dependent MMDTs such as hide, show, fade in, fade out, loop, and stretch. To include a new effect on a multimedia object we have to sub-type the abstract Effect object type. The effects must be executable in real-time. For completion, we define the skeleton of the abstract object type Effect. The Effect_Timestamp object type

```
interface P_MMDT:MMDT {
     attribute MMDT mm_ref; /*Reference to its corresponding MMDT object */
     Relationship CPO belongs_to inverse CPO::multimedia_objects;
     attribute Temporal_Reference p_start;
                         /*Refers to time point on the global play time scale */
                         /*of the CPO referred to by the belongs_to relationship */
     attribute Temporal_Reference p_stop;
};
interface P_PTD_MMDT:P_MMDT {
     attribute Play_Time start;
     attribute Play_Time stop;
     attribute double speed;
     attribute Temporal<Effect_Timestamp, Effect> Effects;
};
interface Effect {
     ....
};
interface Effect_Timestamp:Timestamp {
     attribute Temporal_Reference start;
     attribute Temporal_Reference stop;
};
interface P_Video:P_Stream {
     attribute position x;
     attribute position y;
     attribute float width;
     attribute float height;
     attribute boolean color_mode;
     attribute resolution res;
     ....
};
```

Figure 10.10: Interface specification of temporal issues.

is basically an interval. The Temporal_Reference pointers start and stop must refer to the global time line of the CPO instance that the Effect_Timestamp instance belongs to.

The temporal characteristics of the presentation of all the play time dependent MMDTs has been taken care of by the abstract super-types P_MMDT and P_PTD_MMDT. The object type P_Video is concerned with other aspects of the presentation information for their corresponding MMDT. Fig. 10.11 illustrates how a P_Video object relates multimedia data to a multimedia presentation. In particular, it shows the relationship between start and stop Play_Time attributes, and p_start and p_stop Temporal_Reference attributes of the P_Video object.

## 10.6 EVALUATION OF TOOMM

In this Section, we examine how a lecture given in the electronic classroom can be modeled with TOOMM and stored in a MMDBS (see Section 10.2). We explain the used MMDTs based on an example shown in Fig. 10.12. The Audio object named PMC_Lecture_hour1_clip1 has the smallest LDU_duration, which is the duration of one sample. Hence, it is reasonable to use this LDU_duration as the MTU_duration for the CPO object named Lecture_19_2_1998. The p_start and p_stop values of all the APO instances in the presentation are set

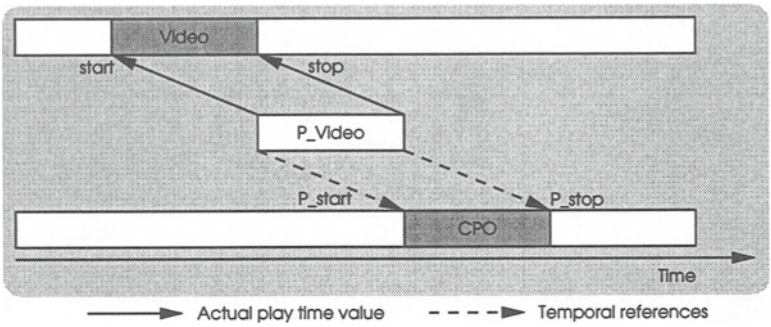


Figure 10.11: P_Video relates multimedia data to a multimedia presentation.

in units of the MTU duration. The Temporal Relationship object TR1 specifies through the Parallel_rel_type attribute that the presentation of P_Video1 and P_Audio1 must start at the same time and stop at the same time. The skew_tolerance attribute specifies that the skew during the concurrent presentation of P_Video1 and P_Audio1 must not exceed 80 ms. The Light_Pen object named Drawing_objects contains two operations which draws a box and a dot during the presentation of the HTML document object called FileSystem. It should be noted that Fig. 10.12 only displays a subset of the objects required to model an entire lecture.

- **Audio and video:** The quality of audio in this application must be very high. The Audio MMDT in TOOMM can be used directly to model the audio in the electronic classroom application. The presentation object type of audio P_Audio can also be used directly. The MMDT Video and its corresponding presentation object type P_Video can also be used directly in this application.

- **Document camera and scanner:** The application program responsible for capturing the events of the lecture can store the data received by the document camera as text if optical character recognition (OCR) tools are available. Alternatively, the data can be stored as a picture using the Picture MMDT. The data coming from the scanner can also be stored using the Picture MMDT.

- **HTML documents (electronic whiteboard):** HTML documents are stored in the MMDBS containing lectures given in the electronic classroom. The HTML document structure is modeled with TOOMM. The electronic whiteboard is the output device for the transparencies similar to a monitor. This implies that the presentation object types of the relevant MMDTs must recognize the electronic whiteboard as an output

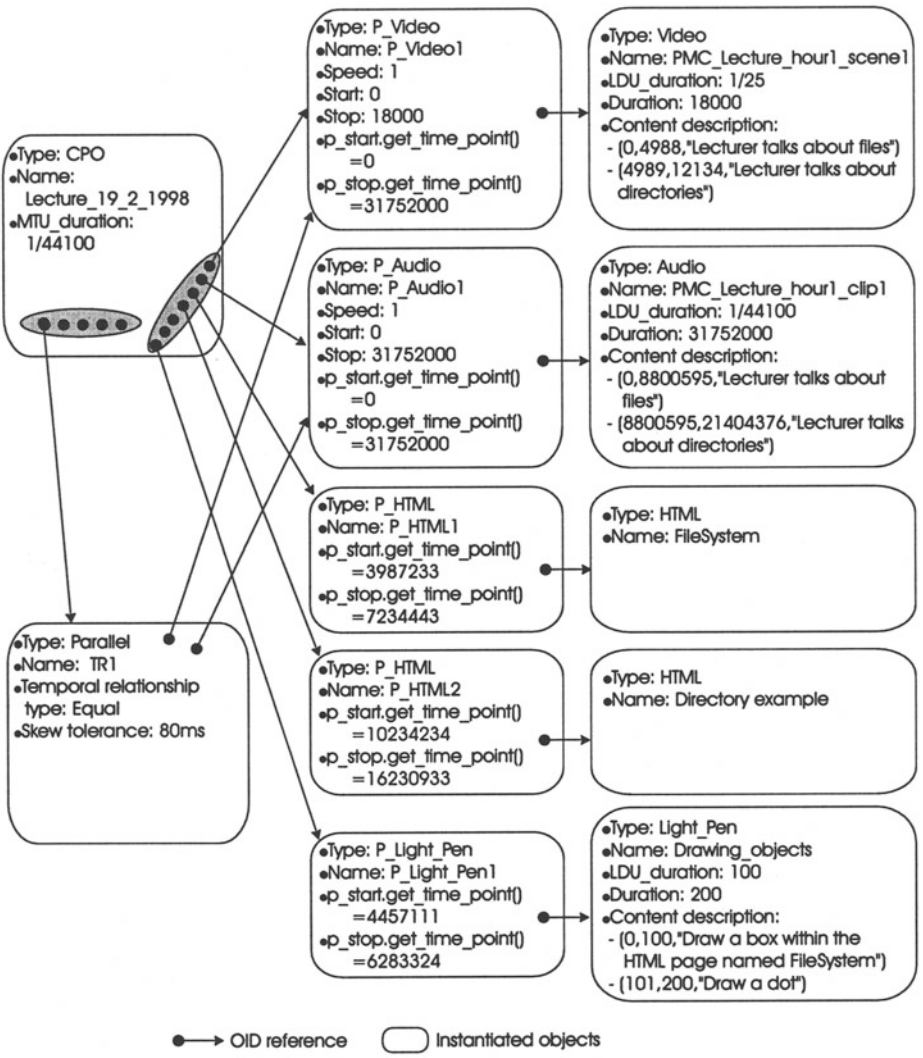


Figure 10.12: Modeling a lecture with TOOMM.

unit. This can be achieved by sub-classing the appropriate presentation object type.

- **Metadata:** The data types needed to model metadata about courses are only basic data types such as numbers and text. These basic data types

can be easily modeled with TOOMM and associated with the complex multimedia objects.

- **Light-pen and other input devices:** The light-pen is used on the electronic whiteboard to issue commands within a browser-like application environment hereafter called browser. We also need to capture what the browser does in response to the actions invoked by the light-pen in order to capture and replay the actions of the light-pen. An application accessing the MMDBS and retrieving a previously recorded lecture must be able to invoke operations, just as if the lecturer using the light-pen issued them in order to properly replay the lecture. During a lecture the actions of the light-pen can also be supplemented with input from the keyboard and mouse attached to the computer running the browser. To model the light-pen, the keyboard, and the mouse using TOOMM, we must find out where the input units fit into the logical data model type hierarchy. Since the actions from the input units happen at irregular intervals with random duration, the CGM MMDT matches the temporal characteristics most accurately.

## 10.7 CONCLUSIONS

In this paper, we present TOOMM, a temporal object-oriented multimedia data model which integrates and extends well known concepts from object-oriented, temporal, and multimedia data models to provide better DBMS support for multimedia applications. We describe the concepts, implementation, and evaluation of TOOMM for the distance education scenario of the University of Oslo.

TOOMM has several advantages compared to other multimedia object models such as a formal structure for representing time in MMDTs, temporal relationships, and structured synchronization information. The separation of presentation information from the actual multimedia data in the SGML/HyTime model lead us to create the logical data model and the presentation model as two separate modules. The logical data model type hierarchy in TOOMM is structured as a tree classifying different MMDTs according to their temporal characteristics. Moreover, each multimedia object in the logical data model can have many corresponding presentation information objects in the presentation model, making it possible to view the data elements in many different ways. Hence, multimedia objects that are used for different purposes need only to be stored once, reducing redundancy and preserving integrity in the DBS. Many multimedia application level QoS parameters are present in both the object types of the logical data model and the presentation model of TOOMM. Temporal relationships contain synchronization requirements between multimedia objects and information on deadlines.

TOOMM provides an advanced framework for creating multiple specialized multimedia presentations based on the same multimedia data without the need for replication. The object types in the presentation model correspond to the

MMDTs in the logical data model. Each presentation object type provides an easy way of specializing the presentation of a MMDT. Sets of presentation object types can be combined with temporal relationships to form complete and highly specialized multimedia presentations.

Adding a new object type to the logical data model in TOOMM requires the developer to choose which of the three categories (PTD_MMDT, PTI_MMDT or Component) the new object type belongs to. However, it is possible that other categories exist which the new object type belongs to, making it worth while to investigate other possible categories and how to include these categories into TOOMM.

Some temporal concepts are not included in TOOMM, but deserve further investigation. For instance, a study on how indeterminate timestamp values can help to model and support user interactions should be performed. The concept of periodicity is also of interest in the context of multimedia data. The presentation of stream MMDTs is strongly periodic, making it suitable for data modeling using periodicity. An investigation on how branching time can help to model multimedia presentations taking alternate routes is also of interest. Another concept that should be investigated is temporal relationships of higher order than two for use with multimedia presentations. Such relationships can model the temporal dependence of the presentation of many multimedia objects, but they can get very complex. In [13] temporal relationships of higher order are investigated, but only for relationships of the same type and not for multimedia applications.

## References

[1] Aberer K., Klas W., Supporting Temporal Multimedia Operations in Object Oriented Database Systems, Proc. of IEEE Multimedia Computing and Systems Conf., Boston (USA), May 1994, pp. 352-361

[2] Allen J. F., Maintaining Knowledge About Temporal Intervals, Communication of the ACM, Vol. 26, No. 11, 1983, pp. 823-843

[3] Bakke, J. W., Hestnes, B., Martinsen, H., Distance Education in the Electronic Classroom, Televerkets Forskningsinstitutt, report TF R 20/94, 1994

[4] Bertino E., Ferrari E., Guerrini G., A Formal Temporal Object-Oriented Data Model, P. Apers, (Ed.), Proc. Fifth International Conference on Extending Database Technology, Avignon (France), March 1996

[5] Böhm K., Rakow T. C., Metadata for Multimedia Documents, ACM SIGMOD Record, Vol . 23, No. 4, December 1994, pp. 21-26

[6] Clifford J., Isakowitz T., On The Semantics of (Bi) Temporal Variable Databases, Proc. of Fourth Int. Conf. on Extending Database Technology, Cambridge (England), March 1994, pp. 215-230

[7] Gibbs S., Breiteneder C., Tsichritzis D., Data Modeling of Time-Based Media, Proc. of ACM SIGMOD Conf., May 1994, pp. 91-102

[8] Goralwalla I., Lentiev Y., Özsu M. T., Szafron D., A Uniform Behavioral Temporal Object Model, University of Alberta, TR 95-13, July 1995

[9] Hjelsvold R., Midtstraum R., Sandstå O., A Temporal Foundation of Video Databases., Proc. of the International Workshop on Temporal Databases, Zürich (Switerland), 1995

[10] Jain R., Hampapur A, Metadata in Video Databases, ACM SIGMOD Record, Vol . 23, No. 4, December 1994, pp. 27-33

[11] Jensen C. S., Clifford J., Elmasri R., Gadia S.K., Hayes P., Jajodia S. (Eds.), A Consensus Glossary of Temporal Database Concepts, ACM SIGMOD Record, Vol. 23, No. 1, March 1994, pp. 52-64

[12] Karmouch A., Emery J., A Playback Schedule Model for Multimedia Documents, IEEE Multimedia, Vol. 3, No. 1, Spring 1996

[13] Little T. D. C., Ghafoor A., Interval-Based Conceptual Models for Time-Dependent Multimedia Data, IEEE Trans. Knowledge and Data Engineering, Vol. 5, No. 4, 1993 pp. 551-563

[14] Özsoyoglu G., Snodgrass R. T., Temporal and Real-Time Databases: A Survey, IEEE Trans. on Knowledge and Data Engineering, Vol. 7, No. 4, August 1995, pp.-513-532

[15] Özsu M. T., Szafron D., El-Medani G., Vittal C., An Object-Oriented Multimedia Database System for a News-on-Demand Application, Multimedia Systems, Vol. 3, 1995, pp. 182-203

[16] Özsu M. T., Peters R. J., Szafron D., Irani B., Lipka A., Munoz A., TIGUKAT: A Uniform Behavioral Objectbase Management System, The VLDB Journal, Vol. 4, 1995, pp. 100-147

[17] Plagemann, T., Goebel, V., Experiences with the Electronic Classroom - QoS Issues in an Advanced Teaching and Research Facility, Proceedings of 5th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'97), Tunis (Tunesia), October 1997, pp. 124-129

[18] Prabhakaran B., Multimedia Database Management Systems, Kluwer Academic Publishers, 1997

[19] Rose E., Segev A., TOODM - A Temporal Object-Oriented Model with Temporal Constraints, Entity-Relationship Conf., 1991, pp. 205-230

[20] Schloss G., Wynblatt M., Building Temporal Structures in a Layered Multimedia Data Model, ACM Multimedia Conf., October 1994, pp. 271-278

[21] Subrahmanian V. S., Jajodia S. (Eds.), Multimedia Database Systems - Issues and Research Directions, Springer, 1996

[22] Tansel A. U., Clifford J., Gadia S., Jajodia S., Segev A., Snodgrass R. T., Temporal Databases: Theory, Design and Implementation, Database Systems and Application Series, Benjamin/Cummings, Redwood City, CA (USA), 1993

[23] Thimm H., Klas W., Playout Management in Multimedia Database Systems, in: Nwosu K. C., Thuraisingham B., Berra P. B. (Eds.), Multimedia

Database Systems - Design and Implementation Strategies, Kluwer Academic Publishers, 1996, pp. 318-376