Universität Stuttgart
Fakultät Informatik

Multimedia Document Handling — A Survey of Concepts and Methods*

Ingo Barth
Gabriel Dermler
Franz Fabian
Kurt Rothermel
Frank Sembach

Institut für parallele und verteilte
Höchstleistungsrechner IPVR
Fakultät Informatik
Universität Stuttgart
Breitwiesenstr. 20 - 22
D-7000 Stuttgart 80

Robert Erfle
Johannes Rückert

IBM
European Networking Center
Vangerowstr. 18
D-6900 Heidelberg 1

Multimedia Document Handling —
A Survey of Concepts and Methods

Ingo Barth, Gabriel Dermler, Franz Fabian,
Kurt Rothermel, Frank Sembach
Robert Erfle, Johannes Rückert

Computer Science Report 5/93
March 1993

* Eine Studie des IPVR der Universität Stuttgart im Auftrag der IBM Deutschland GmbH.

Table of Contents

1 Introduction

1.1 Motivation and scope

Classical document handling systems are among the most frequently used computer applications. Their popularity stems from the implied rise in document quality and handling efficiency. The electronic document at this stage is the exact equivalent of a traditional paper document. It comprises information represented by text and graphical images structured in a desired fashion.

Recent and ongoing technological developments are expected to considerably advance the way in which users interact with their systems as well as the way they communicate among each other. The increase in processing and communications power allows for integrating new media types, such as audio and video, which capture the naturalness of information as experienced by humans in everyday's life. This expressive power will make these new media have an impact in a range of fields, where conveying information to users plays an important role. One prominent field, where a strong impact can be observed, is document handling.

The same technological drive encourages users to cooperate among each other, when working on common work items. As a consequence, document handling will have to support new interaction schemes, which go beyond today's prevalent one-user at a time paradigm.

The development in these two directions will considerably enhance the way documents are handled. These enhancements are the topic of this study. Other current developments concerning documents like the move towards hypertext and hypermedia were not included, although some of the presented techniques apply to these as well. The study is based on evaluating results of recent and ongoing research, including results which have already lead to new document standards. The goal of the study is twofold: first, it was intended to provide a comprehensive overview of requirements and solution approaches and second, to show up issues requiring future research. Three major aspects are treated separately: information structuring, communication support and data management, with the latter incorporating the cooperation-related issues.

1.2 Definition of multimedia documents

The definition both of multimedia and of multimedia documents varies greatly in the literature. While some consider graphics and text already to be a case of multimedia, others use this notion only if time dependent media like audio and video are included as well. In the context of this study we use the term *multimedia* to denote a combination of time dependent and time independent data, leaning on the definition given in [SRR90]. It distinguishes the following basic data types in documents:

text, as the basic representation for written documents,

tables, to give an overview on selected information,

raster graphic, for representation of included pictures,

vector graphic, to support technical drawings,

animated graphic, for the demonstration of moving graphics,

audio, to include sounds and voice annotations and

video, to include moving pictures to a document.

The term *multimedia documents* was coined to reflect the inclusion of multimedia data into documents and the resulting impact with respect to handling support. However, there is no consistent view in the literature, how a multimedia document is to be defined. This study assumes the following understanding:

A document is a self-contained entity of information belonging semantically together. It is stored persistently and may be accessed at any time. For time-dependent media, this means that the data has to be available from a stored file and not from a live source. In addition to pure data, a document comprises structuring information as well. This refers to the content of a document and its presentation. The latter also includes the specification of user interaction during the presentation of a multimedia document.

1.3 Overview of following sections

Section 2 deals with **data modelling and representation** of multimedia documents. Within this field, structuring is an important issue. Different aspects concerning the visual appearance, the underlying logical model and the physical aggregation is discussed. Introducing time dependent content portions into documents, requires specification of temporal relationships, which adds another aspect to the structuring issue. Several approaches are surveyed in order to show how different application areas are reflected by different structuring schemes. The specification of temporal relationships is discussed in a dedicated section, giving a classification of the different concepts.

Another major aspect, discussed in Section 3, is specific **support for communication** as required by time dependent data in documents. This support, referred to as stream-level synchronization, includes the definition of suitable low-level abstractions, procedures for setting up stream channels for multimedia transmission and synchronization control mechanisms.These are discussed in the context of quality-of-service (QoS) parameters which are defined for multimedia communications. Mechanisms are presented either as part of integrated multimedia systems or as stand-alone research proposals. The section concludes with a summary and a short discussion of the approaches found.

Section 4 gives an overview of new requirements for the **storage and retrieval** of multimedia data. The possibility of using traditional DBMS for the storage of multimedia data is evaluated, as well as the necessary improvements to existing DBMS and database access protocols. Several new approaches to store and retrieve multimedia data are evaluated too.

In the part on storage systems we give an overview of the available physical storage systems for multimedia data. We consider the conventional operating system file systems to use them for storing multimedia data. Special file systems for storing continuous media on magnetic disks and multimedia systems with videos stored on compact disks are presented.

Another main point of interest is the area of support for cooperative work, mainly with respect to storage and retrieval of multimedia documents. Users may wish to cooperate in generating complex data objects like multimedia documents. In the part on cooperation we state the requirements for storage systems and applications to support cooperating users. Existing applications in the field of joint editing, like group editors and design databases are investigated, concerning their support for user cooperation.

The **main results** of the previous sections are compiled in the last section. In addition, areas for further research are identified.

2 Structure of Multimedia Documents

2.1 Introduction

When talking about documents, one must distinguish between three different levels or structures.

- The layout structure specifies how the components of a document like titles, paragraphs or figures relate visually, regardless whether the document is to be presented on paper or on screen.
- The logical structure identifies the components of a document and defines logical relations between them, as for example some hierarchy or referencing connection (footnote - footnote reference).
- The physical structure represents the separation of a document into several parts that may be stored separately.

These structures form the main part of a document architecture. Additional considerations cover specification of the basic content types like ASCII text, graphics format and coding, etc. and the relationship between the structures. An example for the latter is formatting, i.e. how logical structure and content will be mapped to the layout structure.

In most document architectures, logical structure and layout structure cannot be distinguished. Thus, the logical structure is inherently expressed by the layout structure e.g. different line spacing for paragraph demarcation or different font sizes for title identification. Clear distinction between the structures offers a greater deal of flexibility concerning handling and processing e.g. different layout specification for the same document.

Multimedia documents extend the paradigm of traditional documents by containing both time independent and time dependent content portions. This requires consideration of temporal relationships between the two. Similar to specifying the spatial relationship of paragraphs, figures and titles when it comes to presentation, now also the temporal relationship between paragraphs, figures, audio sequences and video clips must be specified.

Another issue that becomes more important concerns referencing of external document parts. Video or audio sequences differ considerably from text or graphic content with respect to data quantity and processing constraints. This encourages separation of such content portions from the rest of the document in order to store (and handle) them on special purpose systems, which results in a richer physical structure.

Writing and editing documents is one of the basic tasks integrated in computer applications like office systems, desktop publishing or programming environments. The existence of a plethora of programs and products in this field is quite natural due to the large number of different applications they are used for. This diversity has its counterpart in the diversity of formats and concepts that are used to represent documents. Consequently, the exchange of documents between different products is likely to fail. This problem became the more serious the more computer systems were connected to each other simplifying data exchange in form of documents.

Since a converter based approach between different formats does not fulfil the requirements of openness i.e. free interchange without a priori need to agree on formats or converters people began to think about standardization. This led to two international standards namely ODA [ODA89] and SGML [SGML86]. They aim to enable open document interchange for the office and publishing area. What was said so far, covered traditional documents that contain text, graphics and images. However emerging multimedia data portions in documents will face the same difficulty concerning exchange probably to a greater extent since the formats and concepts used are more diverse. Having in mind the lessons learned from traditional documents standardization is considered in an earlier stage of development. Therefore in this section the focus lies on standards.

The rest of the section is structured as follows. First, basic concepts of data modelling within multimedia documents are discussed. Then the standardization activities in this field are surveyed focusing on a description of the main ideas. A brief introduction of two non-standard approaches within the research community follows. Then the synchronization aspect is analysed classifying standard and non-standard approaches. Finally the section concludes with a short discussion.

2.2 Concepts

Document architectures need mechanisms to specify relationships between certain units of a document. Obvious examples are footnote text - footnote reference relationships, links between chapters (“see chapter ...”) or rules defining which logical portion of the document should be laid out by which layout building block. Generally speaking, there must be means to identify certain document portions (e.g. footnote text) and to specify the semantics or properties of the relationship (e.g. reference).

In document architectures that address the integration of time dependent content portions the mechanisms described above have to be extended considerably. Specification of temporal relationships requires richer means of 1) identification of certain document portions and 2) semantics specification.

1) Unless the smallest document portion to be identified is equivalent to some structural building block (e.g. a paragraph) there must be means to identify parts of building blocks. This may not be straightforward in the case of audio and video sequences.

2) In traditional document architectures the semantics of relationships were mainly references to figures, chapters or entries within the bibliography. But when it comes to expressing temporal behaviour, things become more complex. Two audio sequences may be presented concurrently or sequentially, starting at the same time or with some delay.

For this reason, quite sophisticated mechanisms are required to allow specification of relationships, especially in describing temporal dependencies. In this respect flexibility is an important issue.

Multimedia documents are rather complex accumulations of content portions that belong to different information types. In general, each information type needs dedicated tools that allow for generation, modification and presentation. The object oriented concept of modelling data seems to be a preferred one in the field of multimedia documents, as can be seen both within standards specification and proprietary approaches.

The object oriented model provides means for structuring masses of data into smaller modules which are called objects. They contain so called methods that are used to access and process the object's data which leads to rather encapsulated units. Every unit has an interface supporting invocation of methods. These interfaces may be designed in a uniform manner throughout all objects. Thus handling and processing of different information types is facilitated as tasks like presentation, access or modification lead to present, access or modify calls that are the same for all objects, as the interface is the same. However what is actually performed is different, as the object methods are different. In addition, usage of such autonomous objects enables their sharing among multiple documents. This becomes the more appealing the more effort has been invested in creating the objects.

2.3 Standards

Like in other industries, agreed upon standards play an important role in information technology. They are the basis on which complex applications may be developed without having to design and specify everything from the very beginning. Nowadays developers of computer hardware and software are on the way to define and establish standards that take off the burden of designing and implementing certain modules or formats each time from scratch, but enable at least usage of common architectures, data exchange formats, API's, etc. In the following we will give a survey of standardization in the field of multimedia documents.

2.3.1 ODA / HyperODA

The need to exchange information in form of documents came up early in the environment of office applications. Here the requirement is to enable the "blind document interchange", i.e. the exchange of electronic documents concerning their content and their logical and layout structure without the need to define special agreements in advance. The Open Document Architecture (ODA) addressed this application area in its beginning, but found a wider application area later on. ODA is a CCITT Standards (T.400 series) as well as ISO/IEC (IS 8613).

The ODA standard specifies the logical and layout structures of a document and how they are encoded in ASN.1 (or SGML) for communication purposes. Standardization of the supported content portions is done by referring to the according existing standards, namely text, graphics and image (ASCII and extensions; CGM; G3, G4 FAX) and specifying their interface as content portions to ODA. A processing model for the formatting process, i.e. the generation of the layout of a concrete document is defined as well.

An ODA conforming document consists of:

- logical structure, specifying the document in terms of a document root, composite object and basic object, that can be used by the editing person of a document to structure e.g. a book (root) in chapters and subchapters (composite) and finally into paragraphs, figures, footnotes (basic) parts without bothering about the layout of the parts. To ease the control of the logical structure of a concrete document, a generic logical document structure is specified, that describes the type of a document, e.g. a book consists of chapters and subchapters, a letter consists of a heading, body and ending. The editing itself generates a specific logical structure, that is valid for one concrete document and obeys to the generic structure.
- layout structure, specifying the document in terms of document root, page sets, pages, frames (areas on pages, e.g. column) and blocks (parts of a frame containing the information, e.g. one paragraph, one image). This structure directly establishes the form of the document as printed or displayed after the formatting process has taken place. To enable the mentioned standardized automatic formatting process the layout of a document must be specified, i.e. the rules how a document of a specific type has to be laid out. This specifica-

tion is called generic layout structure, whereas the structure generated by the formatting process of an actual document is called specific layout structure. The latter can be used for conversion into a printer or display format like PostScript[1] or SPDL (Standardized Page Description Language).

- content, that is the basic information carrier and belongs to both the logical and layout structure. ODA is extensible for new content types in specifying content only in terms of an abstract interface to the ODA structures. In general that interface enables handling of content portions during the processing of the ODA structures. This is done by different attributes of content portions concerning all necessary aspects of their participation in logical and layout structures, their editing, formatting and presentation.

Logical and layout structures are associated, since a generic logical object refers to a generic layout object to specify how an according specific logical object should be laid out. The layout can be generated from the logical structure due to the fact that there is a natural order to traverse a document, that is essentially the same in logical and layout structures: Process it from start to end, called the sequential order. The automatic formatting process scans through the specific logical structure of the document, traversing the layout styles defined in the according generic layout objects to generate the specific layout objects.

The logical and layout structures are specified via attributes named *subordinates* to build the hierarchical structure and *layout category* to refer to the generic layout objects. Besides these relationships in ODA documents, it is also possible to refer to bundles of attributes called *styles*, that link the content portions to the basic logical and layout objects, and set user defined references to enable e.g. automatic page numbering.

Besides the already described parts of an ODA document, different specials are standardized in ODA, concerning attributes of the whole document and document transmission in various forms (processable, formatted). So-called Document Application Profiles (DAPs) specify the usage of ODA for specific application areas. A more detailed introduction into ODA may be found in [HKN89].

Including multimedia information into ODA requires mainly the following aspects:

- Definition of interfaces for new content portions audio and video
- Definition of a model for document content layout in time and its integration with the conventional layout in space
- Definition of user interaction for the time dependent presentation of a multimedia document
- Inclusion of non-linear structures into the whole document model
- Extension of the addressing capabilities of the ODA model with respect to partial documents

The described extensions to ODA towards a multimedia ODA are addressed in the standardization arena and are currently in discussion. As an addition to the ODA standard an audio content architecture [SW*90] is discussed. The HyperODA [HODA92] activities have proposed extensions for temporal relationships, non-linear structures and fragmentation of documents. Addressing and referencing techniques are under discussion.

Concerning the new content portions, the handling of their temporal presentation in conjunction with spatial extent layout requires a revised layout process. E.g. audio content has no spatial ex-

1. PostScript is a trademark of the Adobe Systems Inc.

tent, but has a location in a document. This location could be the position between other document parts that do have spatial extent, or the document part, to which audio is linked as an annotation. Similar to an image, video has an appearance in the spatial dimension and in addition a temporal extent.

2.3.2 SGML / HyTime

SGML, the Standard Generalized Markup Language has been designed for usage in environments where the logical structure is of great importance and the layout remains a local matter. The publishing industry and the field of documentation are two examples for such environments. The author is responsible for content and structure whereas the publisher is concerned with layout considerations that usually require high quality standards.

SGML allows for structuring documents by inserting special markup in order to identify logical document parts and provide some relation between them. SGML does not bother about the application semantics of the identified document portions.

The Hypermedia/Time-based Structuring Language [HYTI92] is an SGML application. Thus a HyTime document constitutes first of all an SGML document. Additionally the HyTime standard provides semantic specification governing basic constraints in the field of hypermedia documents.

SGML

An SGML document consists of

- a document type definition (DTD) that provides the specification of a certain document class. The DTD governs the logical and physical structure of the document instance. It defines element types and their relation to each other which forms a hierarchy. The element type names form the basis of start and end tags that identify a portion of a document instance to be of this type. Examples are paragraphs, figures etc. Nesting of elements must always be consistent with the DTD. SGML allows the document content to be of any format or coding. The DTD distinguishes between parsable character data defined by the standard, and so-called non-SGML data that is unknown to SGML. Leaf elements that contain actual data instead of other elements are declared to contain one of the two.

  An element may have any number of attributes that are specified within the DTD. Attributes in SGML are unstructured. Again, SGML does not standardize the attributes' semantics. However two attribute types, namely ID and IDREF play a special role, as they are used to establish connections between elements as for example between a footnote and footnote reference.

- a document instance, that belongs to a certain document class specified by the DTD. The content of a document instance is enriched with readable markup that specifies
  - logical structure = element structure
  - attribute values of individual elements
  - entity references.

An SGML document instance may be subdivided into several pieces, called entities. Entities are internal or external. Internal ones provide a means, by which document parts that appear several times, may be declared once in the DTD and can then be referenced several times in the document instance. External entities work quite the same except that their content appears externally, e.g. in a file. The specification of an external entity in the DTD gives hints of where the content may be found. An entity may also be used as a container of data that is unknown to SGML. In such cases, the entity declaration should contain a notation specification (e.g. the coding scheme of a graphic), which is complemented by a separate notation declaration, intended to tell the application how to process the data (e.g. specifying the program that processes the graphics data).

Entities may be referenced by other entities, which leads to a hierarchical entity structure. It is up to the application to define an appropriate entity structure. In many cases the DTD will be an external entity. This avoids transfer of the DTD provided the receiver of the document also possesses the affiliated DTD.

In SGML the logical structure, represented by the element nesting, and the physical structure, represented by the entity declarations are independent of each other. Thus in principle an entity may contain any portion of the logical structure. SGML knows nothing about layout and consequently does not contain a layout structure. It is the task of a formatting application to provide layout specifications that apply to the elements defined in the DTD. Then every document instance may be formatted accordingly. Similar to this example applications may be defined for certain document type definitions. Further information about SGML may be found in [Gold90] or [Smit89].

SGML is applicable for specification of multimedia documents in a limited manner. As content portions may be of any format or coding, document instances containing stills and audio or video sequences are within the scope of SGML. However there is no concept of relating elements expressing temporal relationships. This is the point where HyTime comes in.

HyTime

HyTime documents support hypermedia applications by specifying basic features that are supposed to be useful in most applications of this type. This includes addressing, linking and alignment in space and time. Addressing aims to identify a certain piece of information within some source. The linking facilities are to establish connections between pieces of information. Alignment specification serves to describe relationships between pieces of information with respect to some finite coordinate system. The mechanisms used are the same, whether axes are measured in time or space dimensions.

HyTime is a construction kit of so called architectural forms. Each form represents a template of an SGML element declaration plus an associated attribute list. HyTime defines what an architectural form is intended for, i.e. its semantic. An application designer first creates an appropriate DTD selecting those architectural forms necessary to describe the intended information. Within a HyTime conforming DTD each element corresponds to an architectural form. Several elements may correspond to the same form as they must not have the same name. Nor do they have to have the whole number of attributes. They may omit ones that are not explicitly required and add private ones. Identification of the associated architectural form is achieved by a special attribute, the HyTime attribute, that carries the form's name. Of course, private attributes require the application to know about the DTD designer's intention, as their semantic cannot be derived from the standard.

Usage of HyTime is not restricted to any specific information format or coding since it only supports generic facilities that may be applied to any information base. Applications do not have to support the complete standard, as HyTime is built of modules. Apart from the base module, each of the other ones is optional and may be ignored. In the following we will present them in turn.

The location address module contains the mechanisms, HyTime offers in order to identify certain pieces of information with respect to the structure of the information base. A coordinate location scheme may be used, if measuring along some coordinate space is feasible. A coordinate space can be a token list e.g. a sentence of words separated by blanks, a tree structure built of nodes and edges, a string built of characters or a finite coordinate space in the more classical sense. A location is identified with a two marker concept, the first marker defining the start of the location and the second one defining the extent. For each case HyTime offers dedicated architectural forms.

If the location to be addressed carries some name, the namespace location scheme will be applied. Finally if both of the former fail, the semantic location mechanism may be used. Locations are addressed by some application specific construct as for example a grep statement on UNIX[1] platforms. In general, location elements are chained, one defining the environment for the next. The last element provides a pointer to the identified portion of information which may then be used to establish relationships, e.g. a hyperlink.

The hyperlink module provides for building link connections. Different architectural forms are supplied, distinguishing among others between (1) contextual links, one link end location is the original location within the data stream, (2) independent links, connecting several link end locations and (3) aggregation links, treating multiple link end locations as one location.

The finite coordinate space (FCS) module allows for aligning pieces of information. HyTime does not know about the dimension of information objects, as it does not know about format or coding. Thus HyTime places bounding boxes within the finite coordinate space, that reference their content. A bounding box together with its content is called an event. Content may be anything from text or graphic pieces to stills, audio and video sequences. Placing is done with the two marker concept mentioned above, either absolutely in coordinate space units or relatively referencing the markers of other events. Alignment in time and space is handled uniformly. A finite coordinate space is built of axes which measure in fractions or multiples of predefined international measurement units like the System International Second (SIsecond) or System International Meter (SImeter). Events may be grouped to event schedules. An FCS may contain any number of event schedules.

For an introduction into HyTime, the following articles may be found useful: [Gold91], [NKN91].

2.3.3 MHEG

MHEG [MHEG92] stands for Multimedia and Hypermedia Information Coding Expert Group. In most cases people refer to the evolving standard as the MHEG standard, while MHEG originally named the standardization group occupied. Currently, MHEG has the status of a working draft. Thus what will be said in the following is due to changes during the future refinement of the standard. Furthermore many aspects are still to be defined.

MHEG aims at specifying the coded representation of final form multimedia and hypermedia information objects. The standard's specifications follow the object oriented model using related terms like class, message and method. However MHEG does not define any representation in an actual system implementation. Therefore realization of MHEG conformant systems may decide to follow a non object oriented approach.

1. UNIX is a trademark of AT&T Bell Laboratories.

MHEG distinguishes on the one hand between basic objects and composite objects and on the other hand between input, output and interactive objects. Basic output objects contain some content and a projector that guides the presentation. Content may be text, graphic, audio etc. Each type has its dedicated basic output class. Input objects are to receive some information from the user. They contain some introduction of what reaction is expected from the user, specification of what kind of input is requested and an input projector, controlling the means by which input will be gained. Composite objects contain basic objects as well as other composite objects. They define relationships between their content objects with respect to hyperlink connections and temporal behaviour. Interactive objects contain both input and output objects and are therefore inherently composite.

Objects are instances of classes. Each class defines a set of attributes (in object oriented systems often called instance variables) and a set of methods that work on the attributes. All the classes form a hierarchy with subclasses inheriting attributes and methods from their parent class. The individual set of attribute values makes up an object instance. Basic output objects have one attribute, carrying the object content, a set of attributes defining an encoding scheme and decoding parameters, both derived from the information type. Attributes of composite objects specify the relationship between all the content objects, e.g. synchronization. Many attributes are not yet defined. The standard provides a basic coding representation for each class using ASN.1 and an alternative representation using an SGML DTD. Of course, both representations are to be isomorphic.

2.3.4 Existing systems

In contrast to the standardization activities mentioned above, many research teams have put considerable effort into making multimedia applications actually work, by implementing non-standard prototype systems. But very few systems contain the notion of multimedia documents. Furthermore, many systems focus on other aspects than data modelling. For this reason, this section remains fairly small.

However, we have chosen two approaches that give some description of how they structure multimedia data. We do not describe the whole prototype system but emphasize on data modelling.

MINOS

MINOS [CTH*86] is a multimedia information system that supports integrated multimedia documents containing text, image and voice portions. MINOS follows the object oriented model in structuring data. Objects belong to classes that are part of subclass and superclass relations. In general, objects may be aggregated to more complex objects, as for example section objects to a chapter object. There are dependent and independent objects. The existence of the former is only possible through the existence of other objects. E.g. a dependent section object has to have a chapter object it belongs to.

MINOS distinguishes between a logical model and a physical model (the physical model may not be mixed with the physical structure as introduced previously in this study). The logical model represents the logical structure of documents, defining object classes like chapter, image or voice classes and their relationships. The physical model describes the presentation structure of a multimedia document on a physical device such as the screen of a workstation. Furthermore there exists a mapping between logical structure and physical structure, specifying which components from one model are mapped to which components of the other. Thus a MINOS multimedia document consists of a logical document instance, a physical document instance and a mapping instance.

A logical document instance may consist of attributes, images, voice sections, text sections and annotations. There may be several relationships between objects of the different parts. A physical document instance contains sequences of pages that may be subdivided into rectangles. Furthermore, attributes may describe presentation properties, as for example fonttype and fontsize, with regard to rectangles or objects, ranking higher in the aggregation hierarchy. A mapping object can tell, where a certain logical object has to be presented within a certain physical object, like a page.

MINOS contains one time dependent information type, namely voice. Voice segments appear in two flavours with respect to presentation. They may either be associated with a voice indicator or may exist as a so called narration. In the first case the voice segment is related to a logical object such as a paragraph or image. The indicator is displayed within the physical object while presenting the paragraph or image. The user may trigger voice reproduction by selecting the indicator. In the second case the segment is related to a page object. Reproduction is initiated by turning the associated page while displaying the document.

Multimedia objects

In their papers [LiGh90a], [LiGh90b] Little and Ghafoor provide a synchronization and storage model for multimedia objects. The synchronization model is based on an enhancement of the Petri net model called Object Composition Petri Net (OCPN). An OCPN corresponds to an ordinary Petri net, enhanced by durations and resources associated with the places of the net. Transitions in the net represent synchronization points, while places indicate processing of some media. Therefore, places have associated with them the required presentation resource (device) and the time required to output presentation data. They showed that the enhanced Petri net model is able to express arbitrary synchronization relationships. (See also Section 2.4.2)

The storage model tells, how the OCPN concept may be mapped to a logical database structure. In order to do this, a database scheme is proposed that implicitly defines the structure of multimedia objects. This structure corresponds to a tree consisting of terminal and nonterminal nodes. Terminal nodes have various attributes describing, among others, the medium type and the location of the data to be presented. Terminal nodes represent the finest possible granularity of synchronization. Nonterminal nodes have a number of attributes, pointers to the left and right child as well as temporal specifications. Within the synchronization model is shown that a binary hierarchical structure is sufficient to express any time relation between multiple objects that belong together. This motivates the two child pointers. The temporal specification relates the child objects.

However, in many cases the binary structure becomes rather complicated and carries redundant information. Therefore the authors introduce meta nodes which are mostly the same as nonterminal nodes, but may contain any number of pointers to child objects. Having multiple child objects, meta objects also have a complex time specification, as all the relationships between the children have to be clarified. With this scheme, multimedia objects may be stored in the database by inserting valid pointers and synchronization specification. The authors also present an algorithm that navigates the database nodes and builds a process tree for presenting the multimedia object.

2.4 Document-level synchronization specification

2.4.1 Introduction

The inclusion of multimedia data into documents leads to the expectation, that presentation of multimedia data can be specified in a way analogous to the specification of layout (presentation) structure of classical text/graphics documents. Unlike text/graphics documents, multimedia information features both spatial and temporal characteristics. While specifying the former is very similar to usual layout characterization, the latter calls for introducing new schemes. It is the latter we focus on in this section.

The presentation of a text/graphics document is defined by the structuring information established prior to its presentation. A user viewing the document may select the parts he wants to view. A multimedia document potentially offers a much greater variety regarding its presentation. Rather simple presentation schemes will only replay a predefined scheme of multimedia data, as is the case, for instance, in a music video combining parts of a song with presenting different imaginative sceneries. The next step is to allow users to make selections of multimedia data to be viewed. An interactive video presentation is a good example for this. Finally, the user interaction may be extended to control the presentation style of multimedia data. Functions like reversing or slowing-down a presentation fall into this category. From these three application classes it can be seen, that the range covered by multimedia documents is obviously very large and exceeds by far the scope of classical text/graphics documents.

Two core abstractions are used to describe multimedia presentation in documents. In the following we use the terminology of [Hoep91a], identifying multimedia objects and actions. Video and audio sequences are examples of simple multimedia objects. Generally speaking, an object corresponds to a presentation which is perceived by a user as a logical unit. Compound objects may be defined out of simple objects, e.g. a movie object consisting of a video and an audio object. The presentation of an object is referred to as an action. Correspondingly, actions can be simple or compound. The presentation of multimedia data is specified in a scheme for scheduling actions, also referred to as presentation scheduling ([Hoep91a]).

Presentation scheduling specifies the sequence of actions which has to be carried out. Besides this, user control of the presentation style can be specified for an action. We subsume under this name all functions influencing the visible temporal behaviour of an action, like reversing presentation, speeding it up, etc.

In the sections below, we present approaches both for presentation scheduling and for presentation style control.

2.4.2 Presentation scheduling

2.4.2.1 Classification of approaches

A scheduling scheme has to specify the actions involved, the generation and the effect of events. The approaches presented below are discussed in terms of these specifications. We classify them into three groups according to the different approaches concerning event generation.

Action driven event generation

In these schemes only two types of events are specified: beginning and ending of an action. The beginning of an action is always triggered by the ending of another action. The ending of an action is enforced inherently by the action itself. In some approaches, asynchronous ending enforced by the user is also possible. The effect of ending events, i.e. the action triggered by them is specified by using different constructs.

In [LiGh90b] the scheduling scheme is given in form of the earlier mentioned object composition Petri nets (OCPN). In these nets, actions are specified as nodes and associated with a fixed duration. The scheme defines 13 elementary OCPNs corresponding to the 13 basic temporal relations which can exist between time intervals including concurrence and sequentiality of presentation. More complex objects are constructed by combining the elementary OCPNs (chaining OCPNs and expanding action nodes). Figure 1 illustrates this approach for a case, where first action A is performed for T_a sec and is followed by the concurrent actions B and C lasting T_b and T_c sec respectively.

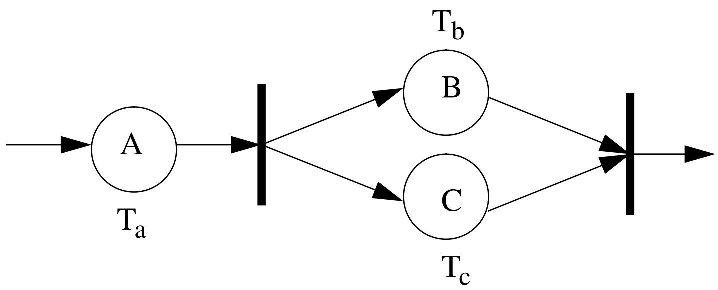

Figure 1: Object composition petri net

Transitions in OCPNs are triggered by (simple or compound) action ending events. In the example above, the ending of A triggers actions B and C. The whole compound action is finished, if both B and C are finished.

A more complex scheme is presented in [Hoep91b] and, less elaborated, in [GBD*91]. Here, path operators are introduced to describe the behaviour of actions. Six basic operators are defined, out of which more complex ones can be composed. For instance, the parallel-last operator specifies parallel execution of two actions, both being started at the same time. The compound presentation ends, when both actions have ended. The parallel-first operator differs from this in that it specifies the ending of the compound presentation as given by the first ending of an action. The sequential execution of two actions is expressed by the sequence operator, the selection operator selects one out of two actions. The repetition of an action is specifiable as well as performing an action a number of times simultaneously. All these operators as well as the compound ones are mappable onto Petri net structures with nodes representing actions with a deterministic or non-deterministic duration. As an example, the Petri net below specifies the parallel actions A1 and A2, started at the same time. Ending of one of these (parallel-first operator ∨) triggers action B1 and ending of both actions (parallel-last operator ∧) starts B2.

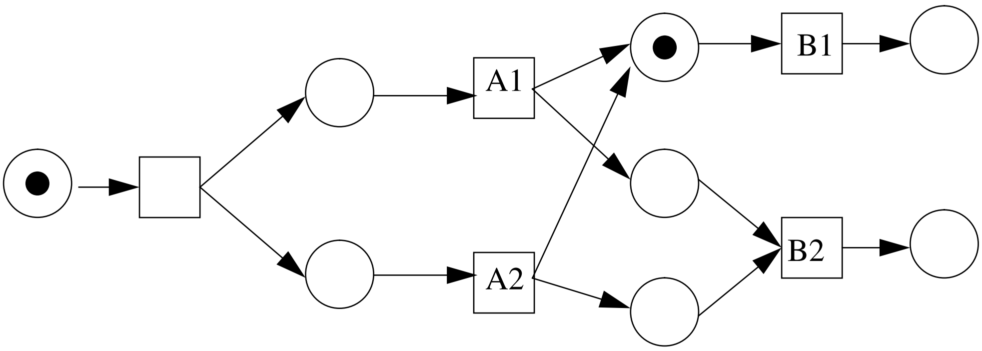

Figure 2: Paths mapped onto a petri net

The scheme expressed by using the basic operators looks like:

path (A1 ∨ A2); B1 end

path (A1 ∧ A2); B2 end.

A third scheme is based on the concept of presentation frames [Hoep91a]. A frame is a unit comprising several actions performed corresponding to a frame type specification. A frame is started by the ending event of one or more earlier frames. More than one ending event can be generated within a frame, again depending on the frame type. Also, a user may enforce the ending both of a single action within a frame and of a whole frame. Action durations can be specified or be left indeterministic. Additionally, the duration of an action can be derived from the corresponding presentation frame by using so-called reconciliation strategies. Five frame types are specified: sequentialiser, parallelizer, splitter, combiner, brancher. Splitters are used to generate more than one ending event, while combiners and branchers specify the start of a frame depending on more than one event.

The three approaches presented above differ in a number of ways. First, the duration of an action is required in the first scheme to be deterministic, while this is not the case in the other schemes. Since the inclusion of user activities as actions implies an indeterministic action duration, the first scheme excludes this possibility. Secondly, the modelling power of the first scheme is rather restricted, while the other two are more powerful in that they can define the start of actions in a greater variety. Furthermore, they are conceived in a modular way, such that later extensions can be added easily. One restriction concerns all three approaches, namely that only action ending events are used to trigger other actions.

Reference point driven event generation

The approach in [BHL91] augments the event generation described previously by allowing for reference points within actions. If the presentation of an object has reached a certain reference point, an event is generated. Reference points are inserted into the objects at arbitrary points. For instance, in a video sequence, events may be inserted to separate different scenes. One may view the concept of reference points as an attempt to avoid handling of objects of too low-level a granularity.

Time system based event generation

These approaches, found in [Gibb91], [BHL91], [AnHo91a], link action initiation to time instants generated by time coordinate systems. Start of an action is simply given as a time. This method is straight-forward, but has two disadvantages: a clock is required for advancing the time system and the scheme does not specify links between actions. While in the approach mentioned first, action terminations initiate other actions, the same is not specifiable here. Hence, the specification relations between actions with indeterministic duration is not possible.[1] Thus, purely time based systems require knowledge of all action initiation times in advance.

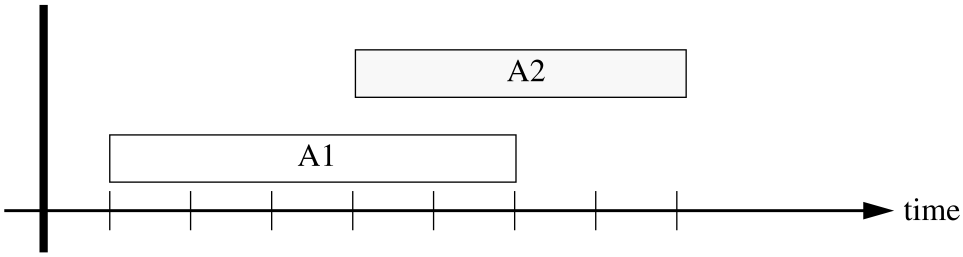

Figure 3: Time system referencing

2.4.2.2 Approaches used in standards

ODA/HyperODA

The HyperODA extensions for temporal relationships are basically the grouping of timed objects, called actions into composite actions. Actions have a start time and duration, possibly influenced by user interaction. Actions trigger other actions by events. Actions of a composite action can be synchronized in sequential, parallel or cyclic order and specify when to deliver an event. Since actions are ordered hierarchically, the description of temporal behaviour is distributed as attributes at each composite action for all of its constituents. This approach is similar to the layout specification in ODA. A separation of the specified temporal relationships into a generic and a specific part is currently not addressed.

SGML/HyTime

Specifying temporal relationships lies in the domain of HyTime. For this purpose, the finite coordinate space module has to be employed. Measuring an axis in some temporal dimension allows to express high-level synchronization by placing events (HyTime jargon for what is called actions here) in a coordinate space. Placement may occur either in absolute terms, e.g. event start at time quantum 4, duration of 5 time quanta, or relative to some other event.

In HyTime axis dimensions may be virtual or real. For both cases placement strategies are exactly the same. An event with a duration of 4 quanta takes twice as long as one with a duration of 2 quanta. Virtual quanta are abstract units that still need some projection to real world units. During rendition, a virtual quantum may be mapped to ten seconds or a minute or some other real quantum. This scheme comes from the music domain, where temporal relationships are

1. This specification may also be desirable for actions with deterministic durations.

specified using full, half or quarter notes. Tempo statements then govern the actual duration of notes. Within the world of music this concept proved to be appropriate for many centuries.

MHEG

In general, an MHEG object is composed of other objects which may themselves be composite. Spatial and temporal relationships are handled almost the same. Within a certain object level, objects may be synchronized either in a parallel or sequential manner. In the first case, the begin of the child objects is determined by some delay with respect to the begin of the parent object. In the second case the begin of a child object refers to the end of the elder sibling, possibly associated with some delay.

MHEG also supports handling of events that may be generated by user interaction or automatically by object processing, e.g. a user stopping an audio recording. For this purpose, the conditional action class is provided. An instance of this class associates an event, possibly including an event condition, with an action to be performed.

2.4.3 Presentation control functions

The range of possible control functions is rather vast and very much depending on the type of document to be viewed and the capabilities of the underlying system. In this section we present control functions found at least in most of the approaches dealing with video.

In [GDT91] four rather obvious control functions with respect to actions are defined: start, stop, pause and resume. Additionally, further medium-dependent functions are included. For instance, "convenient" working with stored video sequences should include "by default" functions like reversing presentation, slowing it down, speeding it up, determining the start position for the presentation or jumping in the presentation. The common denominator of these functions is that they change the visible temporal behaviour of the presentation of an object. One may also say, they influence the behaviour of time (e.g. reversing a video presentation is often referred to as "time running backward"). The solution offered in [GDT91] for facilitating the concept of these functions is to define for each object an object time which is considered as separate from the world time. The latter is supposed to reflect the real time, advancing forever at a constant rate. In contrast, the former is employed to describe the behaviour of the object presentation with respect to the world time.

Four methods for controlling the object time are offered: translate, cue, scale and invert (besides start, stop, pause and resume). A translation positions the start of an action with respect to world time, as described in the previous section. Cue positions the starting point within an object (e.g. 30th second of the video sequence). Scale determines the advance rate of the object time in relation to the world time, while invert specifies the advance direction, i.e. whether the object time should run forward or backward related to the world time. Using these methods it is possible to provide the desired user control.

With respect to presentation control, the discussion within HyperODA has not been deepened. Therefore this aspect must be omitted here.

HyTime does not explicitly include mechanisms or concepts to define user interaction during presentation. However, the basic concepts HyTime provides may be employed in order to express schemes of presentation control. HyTime does not know about action (which in HyTime is called event) content but only its placement. Thus, actions containing user control functions may be used to read control information from the user. Such actions may also be arbitrarily linked to other actions reflecting a different presentation sequence, desired by the user. This is just a very unsophisticated example. The main point is, that HyTime only provides some basic principles that may be employed in many different ways including specification of user interaction.

User driven presentation control in MHEG is expressible by the conditional action class as mentioned above. MHEG provides a set of predefined event types one of which covers user input. With this facility, other actions may be triggered that change the MHEG-object's presentation in the desired way. Apart from that, other event types may be defined by applications.

2.4.4 Discussion

With respect to presentation scheduling, the overall conclusion is that all presented approaches have deficiencies. All of them allow for specifying static presentation schemes, combining actions with fixed durations. Some of them, in particular the time system based ones, exclude linking actions of non-deterministic duration. As a consequence, these schemes do not allow for including user-driven actions into the scheduling scheme. HyTime and MHEG which must also be classified as time system based constitute an exception as they do allow for non-deterministic duration of actions. Secondly, action driven event generation is currently restricted to a rather small set of possible specifications (given by path operator or presentation frame types). Future experience with document handling will have to show, whether the corresponding expressive power is satisfactory.

Unlike presentation scheduling, which is a concept for sequencing and synchronizing actions as a whole, presentation style control influences the temporal behaviour within an action. Controlling the presentation of a multimedia object can be based easily on the separated concepts of world and object time. Controlling the relation between object and world time is a powerful means for specifying the actual temporal behaviour of an action.

Within the standards arena, presentation style control is not covered in detail, as this feature is heavily dependent on the actual application and system platform. For this reason, mainly a framework is provided that corresponds to the concepts of the standard, but still needs further application enhancements.

The actual selection of the control functions will have to reflect both the medium used, and the capabilities of the underlying system hardware.

2.5 Conclusion

None of the document architectures mentioned covers logical, layout and physical structure without any restrictions. This is due to the fact that each architecture has been designed for a specific set of applications which do not require total flexibility. However, with respect to support of new content portions like audio and video, existing concepts are extended. The proposal within HyperODA allows for disaggregation of document instances leading to a richer physical structure. HyTime architectural forms allow for layout specification to a certain extent which introduces some kind of layout structure within the field of SGML based documents employing HyTime.

A second field of extensions covers more flexible and richer techniques for identification and addressing of certain document portions and the establishment of relationships between them.

In this respect HyTime offers a broad framework that also contains provisions for addressing in other document architectures than SGML.

All architectures use the object oriented approach to some extent. In ODA and SGML types of structural building blocks are defined (within the generic structures in ODA and the DTD in SGML) which are similar to the notion of classes, e.g. allow for creation of multiple instances. Neither SGML nor ODA contains the concepts of inheritance and methods. In contrast MHEG completely follows the object oriented approach as do most of the non-standard systems.

Within the standardization of document architectures, two different approaches with respect to the scope of the standard can be distinguished. The ODA standard aims to allow document specification that contains all the information necessary to process the document, e.g. formatting. The receiver of the document does not need to know where the document comes from, which application it belongs to, etc. Therefore every aspect of document structuring and processing has to be standardized like content architectures or layout attributes.

In contrast, SGML is restricted to standardization of syntactical aspects, specifying the logical structure of a document. Users who exchange SGML documents and want them to be laid out in the same way have to agree on a layout specification. Generally speaking, they have to agree on the semantics of the DTD they are using. This practice allows a certain document to be laid out differently by associating different layout specifications. The freedom of defining arbitrary attributes and incorporating any content offers a great deal of flexibility. As can be seen, both standards follow a different concept with none being “better” than the other. The requirements of an application may decide which one is appropriate.

The same principles are valid for support of specifying multimedia documents. Within the ODA field these are the mentioned new content architectures to be defined and the extensions discussed within HyperODA.

On the SGML side, HyTime is intended to support multimedia document specification. Again HyTime only standardizes the basic mechanisms, leaving many issues for the application, as for example format and coding of content portions. Apart from that, HyTime is built of several modules. An application only needs to understand the modules actually used. Thus not every application has to support the complete HyTime standard.

With DSSSL, the Document Style and Semantics Specification Language, a new standard is currently under discussion that aims to allow specification of document processing semantics like formatting. Usage of DSSSL would e.g. enable blind document exchange of SGML documents.

3 Communication Support for Multimedia Data

3.1 Introduction

This section presents mechanisms required to ensure synchronization within a stream (intrastream synchronization) as well as between streams (interstream synchronization). The mechanisms represent the basic level of synchronization which can be applied to multimedia data. Higher-level synchronization schemes, like the ones discussed in the previous section, will employ (and augment) this low-level service.

We use the notion of *stream* to denote a sequence of data units generated at a location and consumed at one or more possibly different locations. The generation and consumption locations are termed stream source and stream sink respectively. We distinguish between continuous and discrete streams. Continuous streams are characterized by a continuous flow of data from source to sink, consisting of data units generated (and consumed) at periodic time intervals. Compressed and uncompressed audio or video are common examples of continuous streams. Discrete streams like usual (text/graphics) program output comprise data units generated at time instants not known in advance. In either case, a constant data unit size is not necessarily presumed, yet some of the mechanisms described below require some kind of knowledge about the data unit size (e.g. the maximum data unit size). In this section we concentrate on support for continuous streams and do not refer explicitly to discrete streams. However, some of the mechanisms described are applicable to discrete streams as well.

Presenting multimedia data to humans is the final intention of multimedia communication. Human perceptions are therefore the base for deriving quality of service values. The parameters used describe the workload involved and the time constraints to be met, such as delay, jitter and skew. The goal of the mechanisms discussed below is to meet these requirements satisfactorily, where satisfactory means both, meeting the time constraints and providing for efficiency in utilizing resources. We refer to [HSS90] for values concerning time constraints.

In the next section, the goals of low-level synchronization are specified and related to mechanisms introduced in later sections. These are presented either in the context of integrated multimedia system approaches or as stand-alone proposals for certain synchronization mechanisms. The last section provides a summary and a short discussion.

3.2 Synchronization goals and their mechanisms

We state the following goals of low-level synchronization: control of delay, jitter and skew, rate matching, start-up synchronization and generation of synchronization events. In the following we explain each of these terms and give an overview of the main groups of mechanisms discussed later in the context of these goals.

Delay is the time between generation and consumption of data. Controlling it means providing means for bounding it and reducing overhead inflicted in processing stages. The former is achieved by reserving required resources at set-up time of multimedia communication and scheduling these according to real-time constraints. Overhead reduction concerns each resource involved, in particular overhead linked with data passing between processing stages (more generally, interprocess communication) and scheduling overhead.

Jitter is defined as the variation of end-to-end delay in a stream. It is controlled (bounded to a desired value) by smoothing out this variation. The means for doing this are buffering and scheduling mechanisms.

Skew is defined for a temporal relationship between data units belonging to different streams. If two data units of two streams are required to be consumed at the same time, but are actually consumed with a certain time delay in between, we refer to this temporal mismatch as skew. Bounding skew is based on the same mechanisms as jitter control. Rate matching is required in order to drive different processing stages at the same rate. Actually, rate matching is a precondition for avoiding jitter, skew or even loss of data units. It is usually based on buffering schemes between processing stages.

Generally, start-up synchronization aims at compensating differences in transmission times before the transmission is actually started. Thus it avoids, as far as possible, initial skew between streams.

Synchronization events are used to support applications in controlling the multimedia communication. Their handling has to rely on certain low-level mechanisms. The way they work is discussed in the subsequent section as well.

The approaches to follow are presented with respect to these goals. Additionally, the stage of connection set-up is presented as a separated functional step. This is done in order to isolate the description of the initiation of the mechanisms provided to meet the synchronization goals.

3.3 Synchronization mechanisms in integrated systems

3.3.1 The DASH system ([AnFe88], [ATW*89], [AnHe90], [Ande90], [AHS90])

The DASH system was developed to study problems arising in large, high-performance distributed systems. Within this frame, support for real-time communication for multimedia was provided. As a basis to this, the specification of the DASH resource model was developed. This model considers a stream connection as made up of a chain of resources. A uniform workload specification scheme is used to specify the temporal characteristics of the data stream exchanged between two adjacent resources. Based on this model a connection set-up procedure was developed. Delay, jitter control as well as start-up synchronization are provided.

Workload specification ([AnHe90])

Each resource can cope with a data stream, the units of which arrive according to a linear bounded arrival process (LBAP). Such a behaviour is described by three parameters: maximum data unit size $S_a max$, maximum data unit rate *Rmax* and maximum burst size *Bmax*. Within any time interval of length t, the number of messages arriving at the resource may not exceed $Bmax+t*Rmax$. A resource generates at its output a data stream which obeys a LBAP specification with unchanged *Smax*, *Rmax* parameters and a possibly changed *Bmax* value. This output LBAP is at the same time the LBAP for the next resource in the chain. Source and sink resources provide only an output or an input LBAP, respectively.

Connection set-up

A stream connection is established by means of a session reservation protocol (SRP) ([AHS90]). A session denotes the use of a specific resource for a specific data stream. Thus, SRP establishes a chain of resource sessions along the stream connection.

The goal of the connection set-up is to provide a stream connection with a requested bounded delay and a sink data buffer allowing for jitter compensation. SRP is carried out by host resource managers (HRM) being able to reserve, free and relax resources. SRP is a two phase protocol, optimizing the distribution of delay among the resources involved. Each HRM performs for each resource a schedulability test, in order to establish the delay bound which can be guaranteed, given the computation load on the resource imposed by stream connections already set up and the load imposed by the new connection. If the client specified maximal end-to-end delay *Em* cannot be granted, no connection is provided and no resources are allocated. If a delay value *Ea* is possible below *Em*, and below a client-specified target value *Et*, the excess delay *Et-Ea* is given back to the HRMs to relax their resource allocations. The relaxation scheme is optimal, given that cost functions are assumed, linking each resource's cost to the processing delay in a restricted way.

The delay parameters computed in the two passes of SRP can be used by each HRM to allocate the appropriate amount of buffer, in order to be able to cope with the specified burst size of the incoming LBAP. Similarly, the computed delay values are used by the last HRM in the chain to reserve sufficient buffer to compensate for delay fluctuations, i.e. to provide for jitter control.

Delay control

The bounding of delay is ensured by the buffer allocations made and by granting a bounded processing time for each resource. The former is done during connection set-up, the latter is ensured by employing scheduling algorithms guaranteeing delay bounds, given the parameters of an input LBAP. Round-robin, FIFO, rate-monotonic and earliest-deadline first scheduling are considered suitable ([AnHo91a]). Bounding delay is further supported by employing efficient interprocess communication between local resources in form of virtual memory remapping, a mechanism which replaces data copying by passing memory area descriptors ([ATW*89]).

Jitter control

Jitter control is performed by each resource by bounding the burst size of the outgoing LBAP or by regulating the burst size of an incoming LBAP to an acceptable level by buffering arrived data units before passing them to the resource scheduler ([ATW*89]). End-to-end jitter control is provided by reserving enough buffer space at the output of the last resource, ensuring that it never runs out of stream data, if it is read out at the nominal stream rate *Rmax*. The buffer size is computed according to the maximal possible delay jitter, given as the difference of the possible maximal and minimal end-to-end delays ([GoAn91]).

Start-up synchronization

A process reading out of the buffer reserved for jitter compensation at the rate *Rmax*, is only granted to always find buffered data, if it defers the start of reading until a certain amount of data has been received. The amount corresponds again to the maximal possible delay jitter.

3.3.2 The ACME continuous media I/O server ([AGH90], [HGA90], [AnHo91a], [AnHo91b], [GoAn91])

The continuous media server was designed as an extension to existing network window servers to support continuous stream handling. The goal is to provide a similar interface for setting-up continuous stream connections between a client and a server as for discrete streams in windowing systems like X, and to provide the mechanisms required for continuous stream data generation and presentation. Several abstractions supporting the specification of synchronization are defined: physical devices, logical devices, compound logical devices, strands, ropes and logical time systems. The underlying low-level synchronization mechanisms, delay control, jitter control and start-up synchronization are presented in relation to these abstractions.

Abstractions ([AGH90])

A physical device (PDev) is a hardware device which is able to generate or consume continuous media data, such as a microphone, camera, etc. Logical devices (LDevs) represent virtual CM I/O devices dealing with exactly one continuous media stream, referred to as a strand. Multiple logical devices may be multiplexed onto one physical device or be connected to many physical devices. The former is the case, for example, for multiple audio streams (Ldevs), which are output (mixed) onto the same loudspeaker (Pdev), while the latter could mean that the same audio stream is output on several loudspeakers. Composite logical devices (CLDevs) represent the source or sink of data on a particular continuous media (CM) connection carrying multiple streams (strands) composed into a CM rope. A rope represents an interleaved structure carrying the data units of the single strands. Ropes are only exchanged between CLDevs. Each CLDev is mapped on (associated with) a number of LDevs sending or receiving the ropes. Logical time systems (LTS) are used to drive the timing of CLDevs. An LTS provides a temporal coordinate system by which the synchronized input or output of several ropes is controlled. End-to-end connections are set up by creating a CM connection to the ACME server, creating the involved LDevs and CLDevs, mapping the LDevs to the desired PDevs and the CLDevs to the desired set of LDevs. Each CLDev is finally associated with an LTS. Starting the LTS results in starting the synchronized I/O of the ropes of the associated CLDevs.

Delay control

Reducing end-to-end delay is supported by reducing the overhead involved in user/kernel mechanisms used to employ system functions, such as CPU scheduling, interprocess communication and I/O. This is done by using a scheduling scheme, termed split-level scheduling and synchronization, and by the use of memory mapped streams ([GoAn91]). The former allows the use of multiple lightweight processes sharing a single virtual address space and a global synchronization of all lightweight processes belonging to the same or to different virtual address spaces. The main advantage of this scheme is that it supports low-overhead communication between processes, while granting correct low-overhead prioritizing of all processes. The latter facilitates communication with I/O devices by supporting user/kernel interprocess communication.

For meeting delay bounds, the deadline/workahead scheduling policy is used. In this, each data unit expected in a stream connection is associated with a logical arrival time (expected arrival time) and a logical delay bound. The sum of the two determines the time by which the processing of the data unit should be finished. At a given time, a real-time process is called critical if a data unit is pending the logical arrival time of which has passed. It is referred to as workahead, if there are pending data units but for none of them has the logical arrival time passed. Critical processes are always given higher priority than workahead ones. Within the same class, earliest deadline scheduling is employed [GoAn91].

Rate matching, jitter and skew control ([AnHo91b], [AnHo91a])

Both intrastream and interstream synchronization are based on the concept of logical time systems. An LTS is a temporal coordinate system which drives data stream I/O according to time stamps carried by data units. These may be explicit or implicitly given by the sequence number of a data unit. The LTS itself (its ticks) may be driven by various sources, such as a physical clock, an I/O device (master device), data arrival on a CM connection. The LTS abstraction specifies local synchronization only, i.e. only presentation or generation of data units at one server site is synchronized.

Jitter control for an output strand is based on driving the LTS by a source with desired accuracy (of periodic time ticks). In an output strand, jitter is smoothed out by employing a strand buffer, in which data is stored before it is presented by a PDev. The size of the buffer is calculated according to the expected maximal delay jitter. This solves the problem, if the buffer has a sufficient size and the arriving rate of data units is identical to the rate, at which the data is read out of the buffer. The calculation of the buffer size is as in the DASH system. Rate mismatch can result in a buffer overrun or underflow. Rate matching is ensured in different ways: by adjusting the presentation rate of the PDev, by resampling the data to match the effective presentation rate, and by skipping small portion of arrived data or by pausing the display of data at following LTS ticks. Jitter control for an input strand is performed similarly.

Output skew control is also based on buffering each involved strand in a buffer and driving the presentation by one LTS. Again rate matching mechanisms may be used. However, if one of the strand buffers starves, it must be decided whether the presentation of the other related strands should be continued or be stopped. Either extreme or an intermediate policy may be selected, allowing for the specification of a skew bound for an LTS. This bound denotes, by how many time units (of LTS) a strand may fall behind, i.e. no data units belonging to that strand are presented. If during this time the strand is able to catch up by receiving a burst of data in the buffer, normal presentation of the other strands is continued. If the skew bound is exceeded, the presentation of the other strands is halted by pausing the driving LTS, until the offending connection has caught up. Similar problems apply to input skew control.

For the handling of ropes, only a kind of rate matching mechanism is specified: Once an input rope (one generated by the server) is initiated, the corresponding CLDev collects data units generated by attached PDevs for component strands (LDevs). The data units are filled into an interleaved structure defined at the creation time of the CLDev (rope format). This format determines the sequence of collecting the strand data units. Rope data units are then buffered in a rope buffer used for flow control towards the client. For this purpose, a buffer level indicator is defined, the near-full pointer. If the client receives rope data via the CM connection too slowly out of the buffer (for example because a flow control at a lower level is used), it is signalled by a corresponding event that a buffer overrun may occur. If it occurs, data units are discarded in earliest-first order. Thus, rate matching between the rope buffer and the client is provided by signalling a rate mismatch to the client or by dropping data units not read by the user in time. For an output rope (to be consumed by the server) the concept is similar with one difference: two buffer level indicators triggering near-empty and near-full events for the client are defined.

Start-up synchronization ([AnHo91b], [AGH90])

Two problems are addressed: Firstly, PDevs may incur different latencies between receiving and presenting data units due to different internal pipelining and buffering schemes. Secondly, data units received on connections may begin arriving at different times on different connections. In both cases, the solution is to buffer data units prior to starting presentation, i.e. prior to initiating the driving LTS. Each PDev is fed with data units, until it signals that it is ready to output these instantaneously, if invoked. To compensate for different initial data arrivals and jitter, a predefined number of data units is to be received on each connection (and buffered either in PDev buffers or strand buffers) before starting presentation. If both preconditions are met, the LTS is initiated and output starts at all PDevs. The number of data units to be received before starting presentation, is calculated from the maximal skew expected on the output connections.

Synchronization events ([AnHo91a])

An LTS is used for specifying (anchoring) synchronization events. These are activated, if the corresponding time is reached in the LTS. If an LTS is linked for instance to an incoming stream or to an output device, an event anchored at time *T* would mean that *T* data units have arrived or been presented. The user may use these events to take different actions, e.g. trigger the presentation of an action corresponding to a scheduling scheme (see Section 3.1), tear-down the connection, etc.

3.3.3 The Lancaster system ([CGHS91], [BHSS90])

This system was designed to support the exchange of continuous streams based on a distributed systems platform. This platform consists of a set of base objects (services): network services, multimedia devices, storage services and the synchronization manager, termed orchestrator. Network services are used to provide a range of protocols to handle multimedia communications. Multimedia devices correspond to logical devices, employing physical devices for I/O and providing an interface for creating communication endpoints and sending and receiving information. Storage services are used to provide a number of specialized storage functions for each medium type. Orchestrator processes are responsible for ensuring synchronization within and between streams. For this, it is supported by mechanisms provided by the network services. Parts of the functionality from above are implemented on a transputer-based front-end unit relieving the main CPU from multimedia data handling.

Connection set-up

Stream connections between distributed sites are established by setting-up processes, linking multimedia devices or storage services with network services. The linking is done by defining the workload imposed in terms of a maximal burst size and burst rate. These parameters are negotiated at connection set-up time between the entities representing the source and sink of the stream. Corresponding buffer space is allocated at both source and sink site of the stream.

Jitter control

Jitter control is basically provided by the burst rate control described above. The burst rate determines the maximally allowed jitter which can be compensated for by the allocated burst buffer. For adapting to jitter variations, burst sizes and the corresponding buffer sizes may be dynamically adjusted by the orchestrator process. This also controls the transmission of a next burst by sending corresponding commands to the sending side or receiving signals from the receiving side.

Skew control

Skew is detected (by an orchestrator process) by monitoring burst transfers in related streams. If any of these is not supplied with data in time, the orchestrator blocks the transmission of data for each related stream and informs the offending stream source to take appropriate actions, such as adjusting burst size or rate. The same actions are performed at the sink of the stream as well. Transmission of all streams is resumed, when he orchestrator is signalled that the corrective actions have taken place. This scheme works for streams merging at a common sink. For streams originating from a common source, the orchestrator is placed at the source and receiving signals from the sink sites indicating when the sinks are ready to receive the next burst.

Synchronization events

An application can define events by specifying in an event script the triggering base service, such as a device or a stream. A script also contains the specification of the actions to be taken on receiving an event.

3.3.4 The Heidelberg system ([HHS91], [HNV92], [Herr92])

As for the last approach, the goal of this system is to realize multimedia communication in a distributed environment. Its architecture defines a set of functional modules, providing for multimedia stream handling. It distinguishes stream handlers, a stream management subsystem, a resource and a buffer management subsystem. Stream handlers are all entities handling multimedia streams, such as multimedia I/O devices, filtering functions, communication subsystems. The stream management subsystem provides means for user control and for enforcing synchronization of streams. The other two subsystems provide support for setting up stream connections and for avoiding overhead in interprocess communication.

Connection set-up

The scheme used is analogous to that described for the DASH system. Connections are established by connecting stream handlers, resulting in a source-to-sink chain. The workload model for data units exchanged between adjacent stream handlers is the LBAP. The SRP protocol is therefore applicable in this architecture as well. For each stream handler, the same operations are performed: schedulability test, resource reservation and scheduling. All these functions are performed by the resource management subsystem. The set-up procedure results in allocated bandwidth and buffer space for each stream handler and the scheduling of threads performing the stream handler functionalities.

Delay control

Reduction of end-to-end delay is achieved by means of a buffer management subsystem. It avoids data copying between stream handlers, where possible. Instead it allows for data passing by referencing data buffers.

Threads are used to reduce communication overhead between (software parts of) stream handlers. The real-time scheduling scheme employed distinguishes 3 priority classes: critical threads, critical threads that have used up their processing time but require further processing and threads that do not represent stream handlers. Within each class, the scheduling is based on the rate monotonic or earliest deadline first algorithm. The choice of preemptive or non-preemptive versions depends on the context-switch overhead provided by the operating system used.

Jitter and skew control

Both is based on (implicit or explicit) timestamping data units and providing for a reference system advancing time. Whenever a data unit has arrived too early, a processing thread (stream handler) is delayed such that the next stream handler receives the data unit according to its arrival specification. Delaying execution of a thread implies buffering of the involved data.

Skew control may also be provided by stream handlers interleaving streams into a compound stream unravelled by a later stream handler in the chain. This ensures that providing for jitter control is sufficient to automatically provide for skew control as well.

At transport level, jitter and skew control is supported by a rate-control scheme.

3.4 Stand-alone synchronization approaches

Connection set-up

Besides SRP, other schemes for setting up real-time channels across networks have been proposed ([HNV92], [FeVe90], [BaMa91]). As far as resource management is concerned, they employ mechanisms similar to those of SRP. In particular, the channel establishment mechanism also requires two passes, the rationale of which is the same as in the case of SRP. The goal in each case is to provide performance guarantees for transferring a given workload across the channel by reserving and scheduling resources.

Jitter control, rate matching

In [Ferr91] a distributed jitter control is employed along a real-time channel. In this scheme each involved node compensates for the jitter introduced by the previous node by delaying the processing of data units, which arrived too early, to the expected canonical arrival time. The approach assumes that no data loss due to buffer overflow is to be tolerated. Under this precondition, the main advantage of the approach is that it reduces overall buffer needs for jitter compensation. In this respect it is superior to a jitter control scheme where jitter is smoothed out only at the destination of a data stream.

A jitter control scheme based on matching the rates of a producer and a consumer of stream data units is presented in [ACA91]. The two entities exchange data via a buffer. The rate of the consumer is derived from the degree to which the buffer is filled with data units, by employing phase-locked loop mechanisms. Three approaches are considered. The first one (direct approach) defines a static conversion table for determining the consumer rate by associating each buffer level with a certain rate value. Level changes are smoothed out previously by a low pass filter. The second approach (charge pump approach) defines two thresholds. If the lower threshold is surpassed, the consumer rate is lowered by a fixed value. Exceeding the upper value causes the consumer rate to be increased. The third approach combines the first two. Simulation results show that the direct approach causes unnecessary fluctuation of the consumer clock, while the charge pump approach tends to cause oscillations between the two buffer limits. The hybrid approach was found to be the best one being able to compensate the weaknesses of the first two approaches.

Start-up synchronization

In [LiGh91] a scheme for initiating the transmission of streams originating at different locations is proposed. It is based on a-priori knowledge of end-to-end transmission delay (which may be deterministic or statistical). In order to start output of data units of all streams at the same time, the stream with the highest delay is started first. The initiation of the streams is delayed according to the difference of their transmission delay and the largest transmission delay.

3.5 Summary

We have presented several low-level synchronization mechanisms, required to provide for guaranteeing user-perceived quality of service parameters expressed as delay, jitter and skew. We conclude this section by summarizing and discussing these.

Connection set-up

End-to-end connections are established by chaining resources to be traversed by data units. Resources must be reserved and scheduled appropriately in order to provide for the requested quality of service. Reservation along the connection is done by means of resource management entities communicating via a protocol which ensures the distribution of end-to-end delay and end-to-end jitter among the resources involved. The protocols are performed in two passes, thus bounding set-up delay to one round-trip. The first pass provides for maximal resource allocation and the second for possibly relaxing or releasing resource allocation depending on whether the requested quality of service could be met or not. Prior to a resource allocation, a schedulability test is performed indicating whether a resource can cope with the new additional connection. Differences in the considered approaches concern the specification of the workload and the type of connections to be established. Workload specification for continuous streams is centered in all approaches around a stream rate, a stream data unit size and an allowed deviation from this periodic behaviour (e.g. burst size, minimum packet interarrival rate), even if the used parameters express these rather implicitly. The type of guarantees for connections set up can be deterministic, statistical or best-effort, corresponding to a more optimistic or pessimistic approach. For each of these types, different schedulability tests are performed. The distribution of end-to-end delay among the resources was optimized in only one approach. It was based on cost functions which were assumed to have a certain characteristic. For other cost function types no optimization scheme was provided.

Delay control

The bounding of delay is ensured by reserving resources and scheduling them in a real-time fashion. Several scheduling algorithms have been proposed ([LiLa73]), from which two are prominently used (in slightly varying versions): rate-monotonic scheduling and earliest-deadline first. Whether preemptive or non-preemptive scheduling is employed, depends on the overhead caused by a context switch. In most cases, priority classes reflect a pessimistic approach, i.e. violating the delay bound in a resource is not tolerated. This may be unnecessary, since the extra delay inflicted may be compensated at subsequent processing stages. Delay reduction is achieved by streamlining resources, i.e. eliminating overheads as much as possible. Data copying is avoided by passing data by reference between resources, if possible, for example by employing threads. Context switch times are reduced by using lightweight processes and providing for a low-overhead communication with the kernel. The latter is also optimized for communication with I/O devices. Finally, non real-time processes are constrained not to inflict real-time processing. For this purpose a real-time and a non real-time environment are created by assigning higher priorities to the former.

Jitter control

Jitter control is based on relating data units to a time system. This timestamping may be implicit if sequence numbers are used or explicit if real-time values are used. Time systems may be driven by different entities, such as physical clocks, I/O devices or data streams. Jitter is smoothed out by delaying processing of data units arriving too early. Buffering of a certain amount of data prior to its consumption ensures that data is available, i.e. data arrives (almost) never too late. The buffer size is to be calculated according to the expected maximal jitter. Adaptions of the buffer size are required, if jitter does not match this value. Distributing jitter control along a chain of resources, reduces overall buffer requirements, if data loss is not tolerated. At the transport system level, rate control is employed to support end-to-end jitter control.

Skew control

The mechanisms used for jitter control, can be employed here as well. Additionally, interleaving of data units (for processing, for transmission) of different streams guarantees that no skew is inflicted, even if jitter may occur. Skew control can be provided for streams originating at the same source and for streams merging at a sink. For both a skew monitoring is employed. Whenever skew is detected, the transmission of all streams is paused until the source respectively the sink of the offending stream has adjusted its behaviour appropriately. However, skew control for more complex configurations as they arise in specific collaboration schemes are not supported in the presented approaches.

Rate matching

Rate matching is a precondition for jitter and skew control. If provided for jitter control, it has to match the rate of data consumption to the rate of data production. The solution can be to vary the production or generation rate or to skip data. Varying the consumer rate is achieved by employing a buffer scheme (software phase-locked loop). The level of data in the buffer is used to adjust the consumer rate. Skipping is done whenever the buffer is overrun. In nearly all approaches, the consumer rate is adjusted and not the producer rate. If provided for skew control, the rates of consumers of the same or different data streams are to be matched. This can be done by varying the consumer rate of the consumers. Alternatively, a consumer may skip data in order to speed up its rate or pause presentation for slowing it down.

Start-up synchronization

This is required for making jitter control feasible and to avoid initial skew of related streams. The former is provided by accumulating data prior to starting its consumption. The latter may be done by flooding the pipeline of processing stages, such that they can begin to deliver data immediately, if they are signalled to do so. Yet, this approach works only for presentation of stored data. If employable, it provides a mechanism for many sources/one sink configurations. A second method compensates differences on different transmission paths, which are known a-priori, by initiating the production of streams at different times.

Synchronization events

Synchronization events are a means to allow higher-level entities, like document handling systems to control multimedia communication. For instance, they support the enforcing of presentation scheduling by signalling action terminations, encountering reference points, etc. Events are prescheduled by anchoring them in a time system, i.e. assigning them a generation time. By attaching a time system to different driving sources, the semantics of event generation can be determined.

4 Management of Multimedia Data

The integration of multimedia data in one document imposes very special requirements on data management in storage systems, that have to support this sort of documents. Section 4.1 discusses data models and storage systems, that are applicable for multimedia documents. Section 4.2 deals with access protocols, that are needed in the case of documents stored in a distributed system. In Section 4.3, data management aspects of cooperative applications in the field of joint editing of multimedia documents are discussed.

4.1 Data model and storage

In this section we give an overview concerning storing and managing multimedia documents. We distinguish between a high-level access to documents through a data base management system (DBMS) and a low-level access to basic multimedia data types provided by the storage system. With multimedia documents the need for storing information of different types must be supported by the DBMS or storage system. New requirements to DBMS and storage systems arise as a consequence of the large amount of document data, and the real-time features associated with continuous media storage and access.

4.1.1 Database management systems

In this section DBMS are considered as one technology, which can be used as a document storage system (DSS). The main focus is on the properties needed to support the storage and retrieval of multimedia documents.

In the following, first a characterization of the application environment and the capabilities of a DSS suitable for the storage of multimedia documents is given. Based on this, requirements to the underlying DBMS are formulated. Next, 3 different approaches for the realization of a DSS are evaluated with respect to these requirements. *Relational database systems* as they are available today are considered as an example for the basic functionality that can be provided. The discussion of an *extensible relational database system* shows the extensions necessary to bring relational DBMS to a level which provides better support for the storage of multimedia documents. *Object-oriented database systems* as an example of a different (and probably more suitable for multimedia documents) database technology are the third approach discussed.

4.1.1.1 Requirements to a DBMS

The requirements to a DBMS used as a base for the realization of a document storage server are different from other database applications like flight reservation, account transactions or the management of employee data. In the second case, the main concern is the efficient storage, update and/or retrieval of large collections of rather small data units. In the case of a DSS, there are bigger but fewer data units to manage. In addition, a user (either directly or by means of an application program) usually is more interested in the retrieval of one or a few specific documents (or parts of them) from the database for further processing than to retrieve a larger collection of stored data.

Clients of a DSS are application programs, which use the server for the permanent storage of documents. Documents can be given to the server for storage, can be fetched by the application for presentation to a user or for external (to the DSS) modification. A modified document can replace the old version or be stored as a new document.

Examples of client applications are:

- Document editing systems which use the DSS for the long-time storage of the documents.
- Information systems which allow a user to select one or more documents for presentation.
- Multimedia mail systems which use the DSS for the storage of received mail and allow the constructing of outgoing mail from stored documents.

Most applications (whether or not they are distributed) using a DSS usually run in an environment consisting of several client workstations and one or more server hosts. Each of these applications needs access to information stored on the server hosts. Therefore, a DSS should support access from other hosts via a computer network. Whether or not the DSS is distributed, is not of essence. For the sake of simplicity, we ignore the case of a distributed server (or assume this to be transparent to a client) in the following discussion. We consider only the simpler case of a single service point. Furthermore, in this section, we concentrate on the interface to the DBMS itself, ignoring the possibility of remote access. The requirements to the communication support, as far as continuous media streams are concerned, were discussed in Section 3. The requirements for remote database access protocols are dealt with in Section 4.2.

Another aspect is the control of heterogeneous storage devices which might be needed for efficient storage of basic multimedia data types like audio and video. Besides magnetic disks and tapes, other physical storage devices for digital or analog data must be supported. In addition, the real-time retrieval of video and audio data needed by some applications requires support at the physical storage level. Characteristics of these storage devices are discussed in detail in Section 4.1.2.

From the viewpoint of an application there are two important properties which have major influence on the quality and range of the support a DSS can give to an application for the storage of documents. These are

- the support for the modelling of data types used in documents and
- quality of services with respect to access mechanisms.

Both of these will be discussed in more detail below.

Support of data modelling

A DSS must be able to store a wide variety of multimedia objects. They range from simple objects, corresponding to basic data types, to complex ones, comprising a multitude of simple objects. The quality of support, a DSS can provide to an application, depends directly on the built-in support for these object types in a DSS.

The support can be considered at 3 levels. Each level is described first from the viewpoint of a multimedia document processing application. This is followed by the description of the corresponding data modelling capabilities that must be supported by the DBMS:

- *The whole document is stored as one unit:* The unit of exchange between applications and the DSS is a whole document. The DSS knows nothing about the internal structure of a document, i.e. it considers a document as an unstructured object. The application is responsible for all processing steps on the document and maintains the internal structure and relationships.

No special support for basic multimedia data types The DBMS must be able to operate on (possibly very large) unstructured data units. Support for variable length data can result in better efficiency.

- *Document parts of different multimedia data types can be stored as different units:* Although the application is responsible for maintaining the structure of the document, single parts can be stored and retrieved as separate basic objects. This allows a much finer granularity in accessing the DSS. Each basic object corresponds to a basic data type, and the DSS can treat objects according to their data type.

Support for basic multimedia data types The DBMS must support all basic data types and operations on them. The support provided, is based on the internal knowledge that different basic data types exist and operations for these are built-in. The degree of support depends on the operations provided. These can range from rather simple ones, like transparent selection of an appropriate storage medium for a data type, to complex ones, like image processing functions.

- *The document structure is explicitly represented in the DSS:* In addition to the support described for the second level, the DSS is able to use and maintain the information concerning the document structure. Examples of structure information are member relationships, sequence of parts, links between parts. Operations provided on these data types allow to construct new documents or different views of one document.

Support for document structure information The DSS must be able to represent one-to-one, one-to-many and many-to-many relations between different objects and provide operations to modify these relations.

Quality of service

There are three basic types of interaction between an application and the DSS:

Extracting whole documents for further processing: A document (or a single document part, stored as a separate object) is extracted from the storage system as a result of a query. The document is copied to the application context, processed and possibly stored back into the DSS. Examples for applications using this scheme of interaction are different kinds of editors (document editors, drawing programs) or multimedia mail systems.

Piecewise or stream-oriented extraction of documents: As multimedia documents possibly comprise huge data volumes, the approach described above can rarely be used efficiently. To reduce memory requirements, it can be more appropriate to retrieve smaller units of data, when they are needed by the application. In this case, retrieval is controlled by the application without further explicit queries to the DSS. Applications which do not need a document as a whole, but operate in a pipeline mode, sequentially processing small data units, benefit from this access method. As an example, such applications may directly present retrieved data to the user with only minimal data processing.

Real-time access to stored documents: The previously described mechanism imposes no real-time requirements on the data extraction process and might be used for discrete media data or small units of video and audio. The methods described above are not sufficient for applications, presenting large units of continuous media data in real-time. Instead, the DSS must deliver data to such an application as fast as the data is needed.

Although the above descriptions focus on data extraction, they apply to data storage analogously. From the three access patterns described, following requirements to a DBMS can be derived:

- Stored documents must be accessible as whole objects through a copying-like operation.
- There must be a stream-oriented interface which allows for piecewise extraction and storage of small data units. Random access (positioning) to the stream might optionally be supported.
- For continuous media applications, a real-time mechanism built upon the stream-interface must be supported.

4.1.1.2 Multimedia database management systems

In this section, three approaches to realize a DSS by use of a DBMS are discussed. The first one evaluates relational DBMS as they are in use today and serves as an example for what is available right now without major modifications to the DBMS. The second approach, extensible relational database systems, is discussed mainly to show the extensions necessary to a relational DBMS to provide adequate support for a wide range of multimedia applications. Third, object-oriented DBMS are discussed as an example of a rather different approach.

Each of these approaches will be evaluated with respect to the requirements to data modelling and access mechanisms stated in the above section.

Relational database systems

Relational database systems together with a query language like SQL are a well understood technology in production use for several years, with users familiar to the underlying concepts. Therefore, it seems to be attractive to use them for the storage and retrieval of multimedia documents and data. But these systems can provide rather limited support only sufficient for a restricted application area.

There is no support for basic multimedia data types, neither in the storage system itself, nor in the query language. Multimedia data can be stored in so-called *long fields* ([HaLo82]) as large unstructured data units of restricted length, but no additional built-in support for the handling of multimedia data types is provided. Size restrictions on data units may affect the usefulness of a relational DBMS for large documents or multimedia data types. The representation of simple document structures and relationships between document parts is possible, but results in the necessity of many tables and complex retrieval expressions ([Meye91], pp. 161-168).

Similar restrictions apply to the access mechanism. Only the most basic method, namely access to whole data units is supported. Although documents can be partitioned into several parts, this must be managed by the application itself and cannot substitute a stream-oriented access mechanism. Finally, the real-time oriented access to stored data is impossible.

Extensible relational database systems

Extending the relational model to incorporate new mechanisms, requires only incremental learning of the new mechanisms by the users, rather than forcing them to adapt to new concepts. But to overcome the limitations of relational DBMS described above, and to support the whole range of multimedia data types and document structures requires substantial modifications and many extensions. The resulting system will be too large, too slow, and too complex to maintain. A solution of this problem is to realize a rather small, but extensible kernel, which can be supplemented with additional components for specific application requirements.

The STARBURST system ([Haas88]) is an example for this approach. To provide adequate support for multimedia data types and structuring information, the addition of new extensible data types suitable for multimedia data, and extensions to the query language with respect to field types are required. Furthermore, support for new mechanisms to allow handling of complex objects which cannot easily be represented within the relational model has to be added.

With these modifications, an extensible DBMS provides the capabilities needed for the representation of all types of multimedia documents, but still lacks the appropriate mechanisms for stream-oriented and real-time oriented access. Only the basic access method (accessing stored documents as whole units) is supported, however with a finer granularity.

Object-oriented databases

As we stated above, the representation of structural information and relations between parts of a document requires substantial modifications to the relational model. Object-oriented systems support these types of information more naturally and give better support for the aggregation and generalization relationships needed for complex nested entities. The object-oriented approach makes it possible to define new data types (together with operations on them) or extend existing data types (by subclassing). This results in better support for the modelling of complex structured objects ([KST92]), the definition of abstract multimedia types and operations on multimedia data units. The support for all three requirements with respect to data modelling is available.

In the area of the access mechanism, little work is done. All systems examined either concentrate on the data modelling aspect or provide integrated I/O modules for the capturing and presentation of multimedia data. The problem of the programming interface providing access to the stored data for further processing outside of the storage system is ignored. Therefore, only some assumptions about the provided possibilities can be made, based on the treatment of direct communication with I/O-devices. The access to whole documents is provided by the accessibility of objects. Stream-oriented access to multimedia data units might be only a minor problem, as this is already provided for direct communication with I/O-devices. Real-time oriented access seems to be the real challenge, because this requires additional mechanisms inside the storage system and additional support by the operating system.

4.1.1.3 Conclusion

Comparing the 3 approaches described above, the object-oriented approach provides the best support for the modelling of both multimedia data types, structural information, and relationships between parts of a document. Support for multimedia data types in relational DBMS requires straightforward and rather simple extensions to the query language and the data definition part of a DBMS. However, the efficient representation of structural and relationship information in the relational model requires new concepts, which are not easy to incorporate.

Providing better access mechanisms, suitable for large multimedia objects and particularly for continuous media data, is easier to add to object-oriented databases than to relational DBMS, due to the inherently better extensibility of the former.

Relational DBMS as available today, provide only minimal support for multimedia document storage and cannot handle continuous media data at all.

4.1.2 Storage systems

Multimedia documents with various basic data types impose a set of requirements to a storage system. Large information structures must be handled to provide storage of pictures and graphics. To present video and audio information contained in a multimedia document, real-time facilities for retrieving these information must be provided.

In this section, we first give an overview of digital and analog storage devices with respect to their usability for multimedia document storage. Then we consider two types of file systems with respect to their ability to store multimedia documents: Conventional file systems and file systems for continuous media. Finally we present two multimedia systems that store multimedia data containing digitized videos on a CD-ROM.

Subsequently, we discuss storage systems with respect to their use in storing multimedia documents. The synchronization of data and management of data dependencies is not considered to be a job of multimedia storage systems. The storage systems are presented under the assumptions, that a multimedia document application uses the storage system for the basic storage service.

4.1.2.1 Storage devices

The storage devices can be divided into two groups:

- devices with digital data storage and
- devices with analog data storage

The analog devices are only used for continuous media data and are attached in a heterogeneous

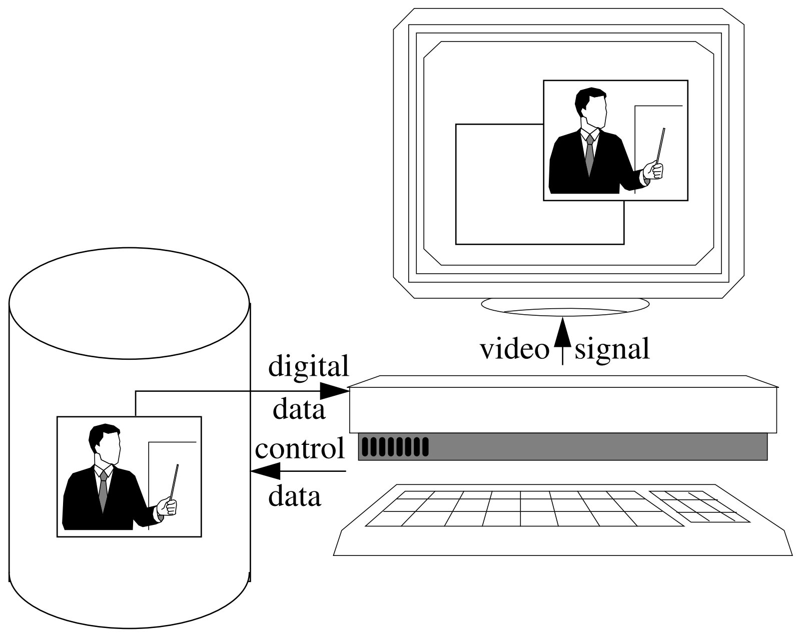

Figure 4: Homogeneous multimedia system

way (see figures 4 and 5). Digital devices can store every type of data in a homogeneous system. Continuous media data is digitized and possibly compressed before storing it in digital devices.

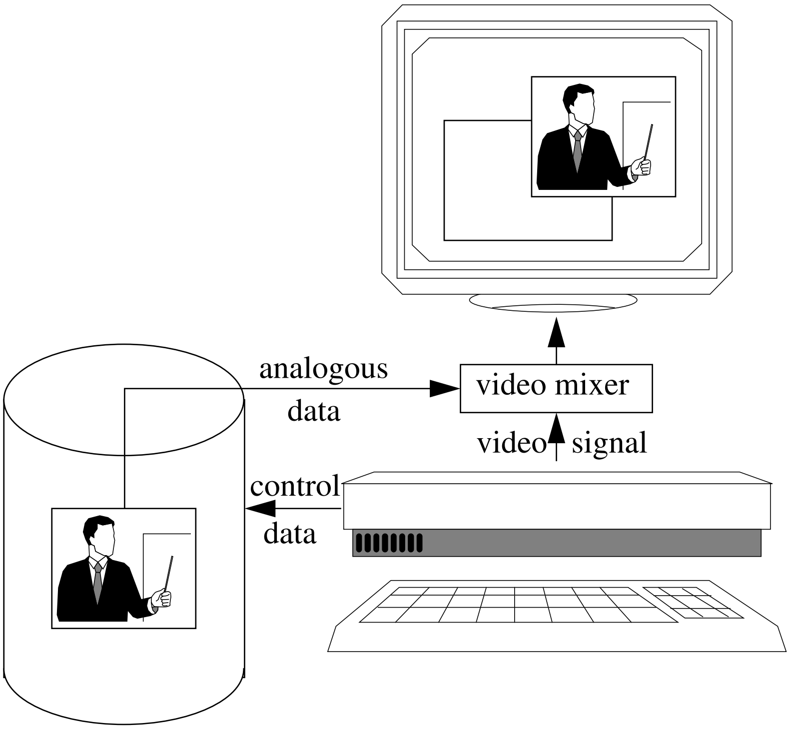

Figure 5: Heterogeneous multimedia system

Digital storage devices

The magnetic disks or hard disks are the common devices for storing mass data with a fast access time. Hard disks use one or more rotating disks with a magnetic material on their surfaces and magnetic heads which are moved in radial direction. The disk rotation speed is constant. The heads read and write the data in concentric circles on the disk. These circles are called tracks. The tracks of all heads together are called cylinders. A track is divided into sectors which are the smallest accessible units of a hard disk.

The access time of a hard disk is the sum of the time needed for moving the heads and the time of waiting for the sector to appear under the heads. In modern hard disk drives we have average access times of about 5 - 30 ms. Another important parameter especially for storing continuous media is the data transfer rate. The data transfer rate depends on many factors, e.g. the transfer rate of the interface, the positioning time, the rotation speed, the number of faulty sectors on a track, and the number of heads in the drive. The transfer rate of today's SCSI drives is about 2 megabytes/second in asynchronous mode and 5 megabytes/second in synchronous mode. These rates give the upper bound of retrievable data based on the hardware. A realistic data rate, which is available over a long period will be in the range of 1 megabyte/second.

With the development of the digital audio playback system for the home audio market, the costs for Compact Disk (CD) players and for production have reached a level allowing optical storage systems to become attractive for being used for digital storage (e.g. for software distributed on CD).

There are three kinds of optical storage devices in use today. First, there is the read-only device CD-ROM (Compact Disk-Read Only Memory). This device was developed based on CD audio music players. The hardware is the same as in the audio CD players. The CD-ROM format is a standard format and an extension of the CD-DA (Compact Disk-Digital Audio) format for recording and reproducing digitized audio sequences. The CD is 12 cm (4.72 in.) in diameter and offers stereo audio play of 74 minutes duration. With 16 bit per sample and channel and a sampling rate of 44.1 kHz the CD-DA stores 764 megabytes of digitized audio. Although the error rate of a CD is in the order of 10^{-8}, this error rate is not adequate for the use as a storage system for digital data. With additional error detection and correction (EDAC) techniques the error rate in CD-ROM is reduced to 10^{-12}. This is equivalent to only one error in every 200 disks. Because of this error correction and the additional information for full random-access the data capacity of a CD-ROM is only 650 megabytes. In CD-DA a digitized audio sequence of 1 second is stored in 75 sectors of 2352 bytes each. In CD-ROM each sector offers only 2048 bytes of user data. This leads to a constant data rate of 150 kilobytes/second. This data rate cannot be changed.

On compact disks the data density is constant on the whole data part. This is achieved by adjusting the rotation speed of the disk when reading information of different distances from the center. The information is written on a spiral track. Therefore the access time for a sector on the disk is the sum of the seek time to physically move the laser head, the synchronization time needed for the laser head to settle and lock on the correct part of the spiral track, and the rotational time for the disk until the desired sector appears under the reading head. As the rotation speed varies, the rotation delay can range up to 300 milliseconds. The average access time is about half a second.

Today there are optical disks which can be written by the end-user. These WORM (Write Once, Read Many) systems can be written, but the information cannot be erased. The WORM systems have lower storage capacities than CD-ROMs, but information can be stored on WORMs for many years. WORMs are produced in many different sizes and use different recording formats. They are not used for publishing informations in a heterogeneous system. The data rates of WORMs are in the range of about 100 kilobytes/second when writing and 600 kilobytes/second when reading. The average access times are in the range of 50-100 ms. The storage capacity depends on the size of the disks and whether the WORM can be used on both sides. For a 5.25" WORM a storage capacity of 400 megabytes per side is reached. Storing information for a long time on WORM disks is better than storing information on tapes, because WORMs keep information without refreshing over at least 10 years and have better access times.

The optical storage systems mentioned above cannot replace hard disks as the storage system in computers, because it is impossible to store any temporary information on them. With the development of erasable and rewritable optical disks, called magneto-optical disks, a competitor for hard disks was born. The magneto-optical disks have a higher data density than hard disks, but they have a higher access time (about 50 - 100 ms) to the data and a lower data transfer rate (about 100 - 500 kilobytes/second, while writing is much slower than reading) than hard disks. The lower access time and the higher transfer rate are the main advantages that make hard disks preferable for the storage of temporal information. The storage capacity of magneto-optical disks is about 300 megabytes per disk and side with ISO-standard format. With non-standard formats, storage capacities of 500 megabytes per side can be reached.

The digital storage systems presented above offer “fast” access to the stored information. Because of the storage in a two-dimensional fashion, every sector on the disks can be reached with a low delay. The presented devices can store and retrieve information contained in multimedia documents. Problems with continuous media having real-time requirements occur, because of the limited storage capacity and the limited data rate. A data rate of about 22 megabytes/second must be provided to handle uncompressed video sequences in VGA quality (640x480 pixels, 24 bit/pixel) at 25 frames/second. The storage capacity of a CD-ROM can keep only 30 seconds of video in this quality. Audio sequences can be stored digitally in all the presented devices, because the data rate of stereo audio data in CD quality is 172 kilobytes/second and all devices can at least support this data rate.

Digital storage devices presented above can be used to store multimedia documents. The fast access and the offered storage capacity and data rate are sufficient for all basic multimedia types in documents with the exception of uncompressed video. Compressed video may be handled by these devices. In Section 4.1.2.4 systems that provide for storage of multimedia data including compressed video on CD-ROM are presented.

Analog storage devices

Storage devices for analog information in multimedia systems are used to record and play video and audio. As these devices present the stored information in a format directly usable for the presentation device, the information is normally mixed with the other presented information, like text and graphics, in analog form under the control of the multimedia system. If the stored information must be manipulated by the computer, a frame grabber device is required to digitize the analog information.

A typical representative of this group of devices is the video cassette recorder (VCR), which stores videos on a magnetic tape. Because of the high data rate (3-5 MHz bandwidth) the video recorder is able to record and play video and audio simultaneously without compression.

Modern video recorders can address single frames with time codes like SMPTE (Society of Motion Picture and Television Engineers). A problem of all tape oriented recording techniques is the long access time when winding the tape. The access time for a video sequence on a video tape can be up to several minutes. Because of this slow access to the stored information, management information like directories, should be kept on other, faster accessible media. A video recorder is a device with exclusive access for a single user.

As the most video recorders store the information in analog form, they are only used for recording videos. Today there are some advances in the home computer sector to use the video recorder for backing up hard disks. This is done by inserting the digital information into a video frame and storing this frame on the VCR as if it were a video. This technique will remain a backup solution for non-commercial computer systems because the existing backup systems, e.g. streamer tapes, are more reliable and are more comfortable in use.

Analog storage devices can only be used to store continuous media data contained in a multimedia document.

Table 1 sums up the properties of the presented storage devices.

Table 1: Storage devices

	Hard disk	CD-ROM	WORM	MO-disk	VCR
access time	5-30 ms	~ 500 ms	~ 80 ms	50-100 ms	minutes
data rate	1 MB/s	150 KB/s	150 KB/s	100-500 KB/s	-5 MHz bandwidth
capacity	-2 GB multiple disks	650 MB	-500 MB 296 MB (ISO) per side	-600 MB per side	-300 min video&audio per tape
rewritability	yes	no	yes (once)/no	yes	yes
text, graphic	yes	yes	yes	yes	no
audio, video	no video	no video	no video	no video	yes
compressed video	yes	yes	yes	yes	-
integration in file systems	yes	yes	yes	yes	only control, no data access without frame grabber

Storage devices in the file system

The storage devices presented previously can be integrated into usual file systems. The disk systems can transparently be integrated into the file system. The only difference a user will see when accessing a CD-ROM instead of a hard disk is, that he cannot write on this device. The integration of a video recorder is less transparent because the information is not stored and accessed in digital form. When digitizing a frame from a video recorder twice, the results may differ for the same frame, because of the noise in the analog information.

With different types of logical devices for the different media, a comfortable multimedia file system can be built. With such a file system, a user does not have to think about the special requirements for the media and the properties of the storage device. The user specifies the type of data when storing multimedia information and the file system uses the corresponding storage medium (see figure 6).

A file system presenting an interface to different basic multimedia data types relieves a multimedia document handling application from the management of the storage system and the special features of storing continuous media.

4.1.2.2 Conventional file systems

In traditional file systems, integrated in non real-time operating systems, the aspect of a fair and similar distribution of the available resources is the main issue. These systems allow every user to start operations at any time. The response time of the file system depends on its actual usage. With continuous media information the need for guaranteed data rates and access times arises.

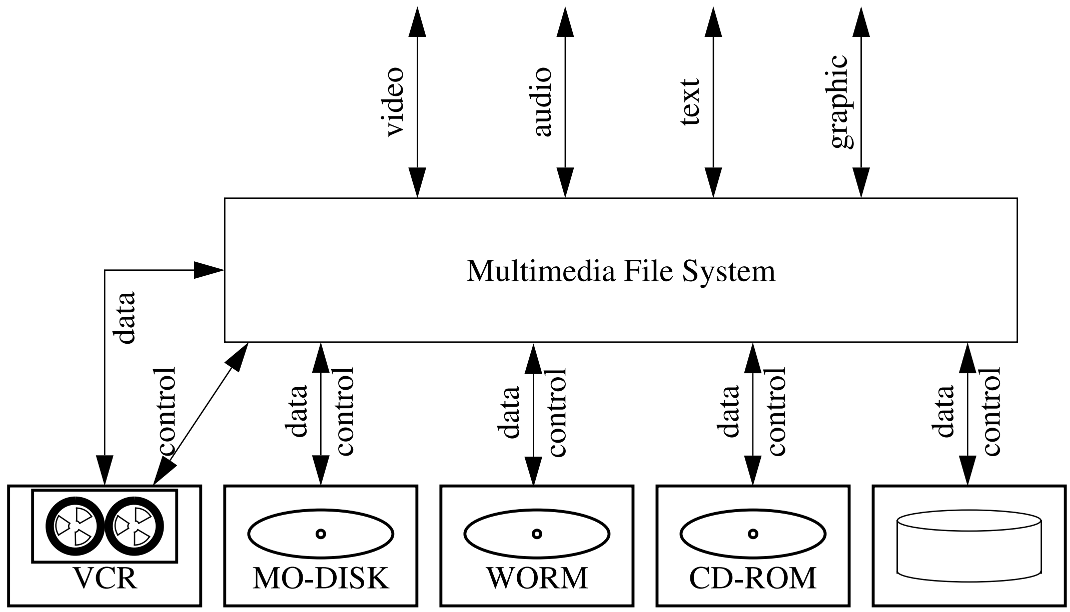

Figure 6: Multimedia file system

Because of the unpredictability of the data rate in these systems, there is a problem concerning the data retrieval of audio or video information with a long duration.

Conventional file systems have no typed files (except directories as a special case of a normal file). If the information should be structured, the user has to use directories to group files of one type or file name extensions to mark a file to be of the desired type. Some file systems offer file types for tabular information (ISAM). The information stored in a file is not recognized by the file system. There is no support for type dependent handling of the files in a conventional file system.

As file systems offer file oriented access to the stored information, they can be used for the storage of text (one text per file), of graphics (one picture per file) and for short audio sequences, which can be buffered in main memory.

Traditional file systems do not provide for the storage of continuous media with its special features. Portions of continuous media streams must be retrieved of periodic time intervals. This can require the storage of information in special patterns, which allow the hardware to provide for the right timing when accessing data.

Conventional storage systems can restrictively manage multimedia document storage. Texts, graphics and still videos can be managed without any problems. Storing of continuous media like audio and video sequences in such systems depends on the application presenting the multimedia document. If such an application accesses video and audio data as one information block and presents it on its own, then conventional file systems are sufficient for multimedia document storage. If the application requires a continuous media stream synchronized to the presentation device, conventional storage systems cannot handle continuous media in multimedia documents because of the missing real-time properties.

4.1.2.3 File systems for continuous media

In this part we take a look at existing approaches to construct a storage system for continuous media on hard disks. These approaches are considered with respect to the required parameters and the offered functionality.

With the development of multimedia devices for computers which convert video and audio into a digital representation, the problem of recording and playing back continuous media has arisen. The storage of continuous media on conventional storage devices, such as hard disks, presents new problems in the management of these devices.

The management must deal with the disk layout, the scattering of information on a disk, in order to provide a continuous read or write. If the time between two successive information blocks for the manipulation or presentation is the same as the time between these blocks when retrieving them from disk, the hardware condition for the retrieval of continuous media is provided.

To provide a file system with more than one continuous media stream, the scheduling of the disks for access by different streams must be included into the scheduling plan. Supported by a buffering scheme, a strategy for minimizing the head movement of the disks can be realized.

A video is typically combined with an audio sequence in synchronous way (lip synchronization). Normally a video and an audio stream have different data rates. There exist two different solutions for storing synchronized streams with different data rates. First an interleaved data stream consisting of the video and the audio stream can be built. This solution requires computation power for interleaving the streams, but makes it easy to handle the streams synchronously, because of the preserved inter-stream synchronization. Some video compression systems combine audio and video into one data stream. This composition is done in specialized hardware, that does the compression or decompression (e.g. DVI[1] [Luth91] and MPEG [Gall91], [StEd91]). This technique limits the reusability of single stream sequences, as they cannot be accessed separately. The other solution stores each stream separately. This has the disadvantage that explicit synchronization between the streams is required. The advantage of this storage scheme is, that we can retrieve other combinations than those stored, e.g. for a multi-lingual video synchronization, and that the storage for each media can be optimized. As each stream is accessible separately, any stream sequence can be use to build new compound sequences. To provide a flexible mechanism to store continuous media streams, both methods should be provided by the file system on demand of the application.

The management and size of buffer space required for continuous media streams and the granularity of information is an issue for research.

As the information volume of continuous media is potentially huge, the system should prevent unnecessary copying of information. To make the stored information editable, a subdivision of the whole information into raw continuous media information and management information is done. The management information contains pointers to the raw information (sound samples or video frames). This information can be changed easily by updating the pointer information. This method of storing continuous media was first presented in the voice manager of the Etherphone system.

The Etherphone system integrates a telephone functionality into an LAN environment, is based on the Cedar system. The voice manager [TeSw88] provides storage for voice, telephone conversation and other sounds recorded at reasonable fidelity. The storage in the voice manager is

1. DVI is a trademark of Intel Corporation.

based on the raw sound samples called the VoiceFiles. The information contained in the VoiceFiles must be captured and played back of 64 kilobit/second. A special purpose extension of the standard file system was built to assure this real-time feature. A sequence of stored voice samples is called a VoiceRope. VoiceRopes contain the information of the sequence and the parts of the VoiceFiles that form the stored audio sequence.

The storage of continuous media on hard disks is a new research field. In [AOG91] and [RaVi91] the problems in designing file systems for continuous media are described. [AOG91] presents simulation results of a research prototype, called the continuous media file system (CMFS).

File systems providing support for continuous media storage are a necessity for multimedia document storage. Without such file systems, only limited use of continuous media in multimedia documents can be provided.

4.1.2.4 Existing multimedia systems with videos on CD

Among todays multimedia systems there are very few that offer video stored on digital devices. There are two systems based on CD-ROMs that offer digitized audio and video in addition to the usual media. To store a video on a compact disk, the data rate must be reduced down to 1.2 megabits/second. With compression algorithms like MPEG and DVI such a reduction can be achieved. The quality of the video is not the same as the quality of still pictures on computer screens, but it is sufficient for the purpose the video is used for in many applications.

DVI system

The DVI system is widely used in the world of the IBM PC/AT[1] and compatible machines. The DVI system consists of the DVI hardware, placed on plug-in cards for PCs and the DVI software. The DVI software can be distributed on CD-ROMs. There are a lot of applications available for DVI systems ([Ripl89], [Luth91], [HKL*91]).

A CD-ROM for a DVI system can store up to

- 650 000 pages of text or
- 5 hours of audio in FM stereo quality or
- 5000 video stills with very high resolution (1024x512) or
- 1 hour of full screen and full motion video with companion audio.

The full screen, full motion video has approximately VCR quality and a resolution of 256 x 240 pixels with 30 frames/second. To get this data rate, a data compression to a value of 0.66 bits/pixel must be achieved. This compression is too complex to be performed in real-time. The so called product-level video (PLV) is produced from a video tape. The compression of one frame takes about 3 seconds, which is still 90 times slower than real-time.

A real-time video (RTV) compression is supported by the DVI system, but as the compression factor is not as good, the resolution of the frames is only 1/4 of PLV's resolution and only 10 frames/second can be stored with the CD-ROM data rate. With a higher data rate a maximum of 15 frames/second can be reached.

1. IBM PC/AT is a trademark of International Business Machines corporation.

CD-I system

CD-I (CD-Interactive) was created by Philips and Sony and is a complete multimedia system based on data storage on CD-ROM. The multimedia system can be used as a stand alone system or it can be attached to another computer which controls the multimedia hardware. ([IMS92], [Sijs91])

A CD-ROM for CD-I systems can store up to

- 250 000 pages of text or
- 2 hours of HiFi sounds or
- 7000 full colour still pictures or
- 72 minutes of full screen and full motion video with companion audio.

The CD-I hardware is based on a CPU of the Motorola 68000-family and has custom VLSI chips for video and audio handling. The hardware offers different interfaces to connect e. g. a pointing device, a keyboard, an external video source or another computer. Video and audio presentation is done by the CD-I system. There is only an analog interface to this information. The CD-I hardware is able to play conventional audio CDs and to display pictures stored on Photo-CDs.

The video chip can do full motion video decompression based on a subset of the MPEG ISO standard [Gall91] and can present still videos in different encoding schemes. The audio chip offers different audio quality levels that reach from CD-DA quality (100% of data rate) to an AM stereo quality (12.5% of data rate). Up to 16 mono or 8 stereo audio channels can exist simultaneously in the same CD-I track ([Meiß91]).

The CD-I system is running the real-time operating system CD-RTOS, which is an extended version of OS-9. The CD-I system can be programmed by downloading a program into the system's memory or by inserting a CD with the application code on it. The CD-I system has two programming interfaces on different abstraction levels. The native CD-I API (NAPI) provides the elementary functions. The high level API (CD-I API) is available on top of the NAPI.

Because of the programmability of the CD-I system and its great number of interfaces, a non multimedia workstation or PC can get multimedia features by adding a CD-I system to it. This is a heterogeneous form of including multimedia features into a workstation.

4.1.3 Conclusion

Todays storage devices are capable of storing multimedia documents with their basic data types. Storing video information is only supported when the information is compressed or only accessed in analog form. The other basic multimedia types can be stored with or without compression on digital devices.

Storage devices are usually supported by file systems. Conventional file systems support the storage of multimedia documents with the exception of continuous media with real-time requirements. Extending conventional file systems with features for continuous media support and a typed interface for different data types should be offered to support multimedia document storage by a file system.

4.2 Access protocols

In this section, two standards for remote access to database or document storage and retrieval systems are discussed. These two are Remote Database Access (RDA) for remote DBMS access and Document Filing and Retrieval (DFR) for document storage and retrieval. Both are located in the application layer of the ISO/OSI hierarchy.

As both protocols deal with the interface between the application and the storage system, the discussion will focus on the requirements for the quality of services as they are formulated in Section 4.1.1.1. In short, these requirements are:

- Stored data must be accessible as one data unit by a copying-like operation.
- There must be support for a stream-oriented interface which allows the piecewise extraction and storage of smaller data units.
- For continuous media applications, a real-time capable mechanism build upon the stream-oriented interface must be supported.

4.2.1 Remote Database Access

The RDA standard has 2 parts, the first (ISO/IEC 9579-1: Generic Model, Service, and Protocol, [RDAG90]) defines generic RDA operations with generic parameters, the second (ISO/IEC 9579-2: SQL Specialization) defines special SQL dependent types for the parameters of the RDA operations and makes some further restrictions.

RDA, Part 1, defines 5 categories of operations for the RDA service, namely *Dialog management*, *Resource management*, *Execution of remote operations*, *Transaction management* and *Control services*. The discussion focuses on the execution of remote operations with respect to multimedia data retrieval. The other categories are not directly involved in the data transfer process and can therefore be ignored.

There are 2 possibilities to send queries to an RDA server:

- immediate execution (R-ExecuteDBL operation): specify a DBL query, send it to the server, which will execute the query immediately and send back the result to the client.
- stored execution (R-DefineDBL, R-InvokeDBL): specify a DBL query, send it to the server, which stores it. Later, the client can invoke the stored query several times with different arguments.

Although there are some minor differences in the concrete specification of the parameters, both methods have enough in common (e.g. the structure of the argument and result parameters) to suffer from the same problems with respect to multimedia data, especially continuous media data.

The RDA standard defines the result of a remote query as a structured data type (specified with ASN.1), which contains among other fields a list of result values. The exact meaning and structure of one of these result values is defined in a specialization. For SQL, for example, this list of values represents a list of tuples, each containing one or more elements. Some of these elements in turn may contain large pieces of data representing multimedia information.

The first requirement stated above can be fulfilled with RDA, although there might be restrictions with respect to huge data units, depending on the implementation of RDA and the lower communication modules it is based upon. Part of this problem can be solved by partitioning the data in smaller pieces, which are stored as different units in the DBMS. This solution is not transparent for the application. To get all parts from the DBMS server, the application must explicitly invoke several queries. The overall need for temporary buffer space can be lowered with this approach, but the complexity of the application, the overhead for additional communication and the resource requirements at the server side will increase. Especially the response time is affected, as RDA allows only one outstanding request, so batching of several requests to the server is not possible.

RDA is based on the Remote Operation Service Element (ROSE, [ROSE89]) which provides **no bulk data transfer service** to the service user(s). Therefore, the support of a stream-oriented access method, which allows the transparent transfer of data in smaller units, is not possible. The same holds for the real-time transfer of continuous media data too, which could be realized on top of the bulk data transfer service. But even the addition of a bulk data transfer service to the RDA (and/or the ROSE) services is not enough, because there exists **no isochronous transport protocol** in the ISO/OSI hierarchy. Furthermore, both RDA and the underlying communication module protocols don't support the necessary Type Of Service parameters (guaranteed bandwidth, maximum delay) to provide adequate service for the timely transmission of continuous media data.

4.2.2 DFR

The DFR standard (ISO/IEC 10166, [DFR91]), defines a service for the remote storage and retrieval of documents in a client-server-environment. Each document has associated attributes, which can be used to support search for documents in the document store or are used for version and access control. Beside the information stored in the attributes, the server has no knowledge of the contents or internal structure of the stored documents.

One of the defined operations, READ, allows a client to access the document itself and/or the values of the attributes. The document is sent as one piece of data from the DFR server to the DFR client and then forwarded to the application.

The DFR access protocol is defined through the specification of data structures and the usage of supporting services like ROSE and ACSE. Although DFR is different from RDA in functionality, both share some basic principles. Both are based upon ROSE and data transfer is in the form of atomic operations, which contain the whole document as one piece of data. Therefore the same problems with respect to multimedia, which are described in the previous section for RDA, exist for DFR as well. These are, as above, the lack of

- a stream-oriented bulk data transfer service, which supports the transfer of large data units in smaller pieces
- support for isochronous transport service and interfaces for the specification of quality of service parameters, needed for the real-time transfer of continuous media data.

4.2.2.1 Conclusion

Access to remote MMDBMS based on either RDA or DFR provides only limited functionality. Both RDA and DFR are partially capable of supporting the copying-like access method sufficient for some multimedia applications as long as the data units are not too large. Stream-oriented and real-time access might only be provided through massive extensions (addition of bulk data transfer and Quality Of Service parameters) to these protocols and requires also extensions for isochronous transfer of data in the supporting protocol layers below of them.

4.3 Cooperation

Computer support for cooperating users is a relatively new research field. In conventional applications, measures were taken, to allow concurrent access to shared data by multiple users. In the database context, the notion of a transaction was introduced. ([EGLT76], [Gray78]) A transaction is an atomic operation on shared data, i.e. it is either executed as a whole or not at all. A multi-user system, based on the traditional transaction mechanism, shows the behaviour of a single-user system to all users. As a consequence, cooperation of users cannot be achieved in these systems. In conventional database applications, transactions typically run for a short time and access a small amount of data. In this context, the consequence of a transaction abort due to a system crash, deadlock, or other failure is negligible.

In applications working with complex data objects, long-lasting operations on these objects should not be modelled as conventional transactions, as the abort of a transaction which ran for some time, may result in the loss of a considerable amount of work. Even worse, system crashes and deadlocks probably affect long-lived transactions more often than short ones. Examples for applications handling complex objects and using long-lasting operations are found in the field of design environments like CAD systems, software-engineering environments and publication systems.

To support this sort of applications, new transaction concepts have been developed, in which transactions are composed of a hierarchy of subtransactions ([Moss81]). In these nested transactions, the failure of parts of the transaction does not have to cause the whole transaction to abort, alternatives to the failed operation can be tried instead. An extension of nested transactions are multilevel transactions and, as a further extension, open nested transactions, introduced by Weikum and others ([WeHa91], [WeSc92]). A subtransaction of an open nested transaction may commit and therefore make its results visible to other top-level transactions, before its own top-level transaction commits. In open nested transactions, the abort of a top-level transaction requires compensation for committed subtransactions.

The applications mentioned above are typically used by several users or groups of users, who want to cooperate to reach a common goal, namely to design a potentially large object, consisting of multimedia data. In the following section, we will state the requirements to the data-management system, which result from the cooperative use of multimedia data. User cooperation roughly relies on the same technology, that makes handling of continuous media data practical, e.g. powerful desktop workstations and high bandwidth networks. As both research fields are relatively new, most approaches to user cooperation on shared documents are not concerned with the integration of continuous media. Nevertheless, the requirements imposed on data management by cooperative handling of discrete media documents, will occur in the same or an even stronger form, if continuous media are taken into account.

4.3.1 Requirements

The applications, that were investigated for this report under the aspect of cooperation, are in the field of joint editing. Examples are Computer Supported Cooperative Work (CSCW) group editors, CAD- and CASE-systems, and publication systems. These applications have the following common properties, where data management and cooperation are concerned.

Large and complex data objects. The applications handle large and complex data objects like CAD-objects, publications, multimedia documents and software projects. All these objects consist of components of different media like text, raster- and vector-graphics, though continuous media play only a secondary role in most existing systems.

Long lasting operations. As all these applications have interactive user interfaces, manipulations of objects may last for a considerable amount of time, typically hours to days. As a consequence, most applications manipulate data in a read-modify-write-cycle, which means, that they do not work directly on the data managed by the storage system.

Multiuser cooperation. A typical scenario in these applications are multiple users who cooperate to design a complex object. In contrast to concurrent use of shared data, cooperating users have a common goal and know about each other's tasks. Therefore they might be interested in viewing and discussing intermediate results of their peers.

Resulting from these common properties, the following requirements can be deduced for the synchronization and recovery mechanisms that are needed in data management systems supporting cooperation on multimedia data. (In the context of Section 4.3, the term *synchronization* always refers to the synchronization of access operations on shared data.)

High availability of data. Cooperating users should be able to access shared data as freely as possible. Blocking each other's work must be reduced to a minimum.

Fast propagation of changes. Primarily in applications, where multiple users work on the same data simultaneously, all users should be able to see the changes made by others as soon as possible. This requirement appears in an extreme form in applications which work on a *What You See Is What I See (WYSIWIS)* basis, where each participant in a cooperative session can see the same screen image.

Application-specific correctness criteria. Serializability is the traditional correctness criterion of transaction based systems. In these systems, operations are executed in a way which is equivalent to a serial execution. As this restricts cooperation in a considerable way, cooperative applications must be able to define own correctness criteria which take into account the semantics of the application.

Minimal amount of lost work. As operations on complex objects can contain a considerable amount of work, synchronization and recovery have to be flexible enough to preserve as much of this work as possible, when a failure occurs.

Partial undo. As cooperating users can commit single steps of operations on complex objects, they also need the possibility to undo these steps one by one. Recovery has to take into account, that operations of other users may depend on the steps which are undone. Therefore user interaction may be required to perform correct recovery actions.

Multilevel synchronization. In a typical cooperative application, three levels of data copies can be identified, which differ in persistence, up-to-dateness and form of usage: the data on the storage system, in the application workspace and on the user display. In a typical setting, the persistent data in the storage system are copied to the workspace of the application, where all operations on the data are performed, before they are written back to the storage system. Parts of the data in the application are converted to a human readable form and displayed to the user. The working-copy of the application is the most up-to-date version, but exists only in volatile storage. The information presented to the user is redundant and can always be created from the application workspace. The persistent data on the storage system can be out-of-date, compared to the working-copy of the application. A storage system that has to support cooperative applications using multimedia data, has to provide means, that allow synchronization on all three levels of data-copies: Persistent data, application workspace and user presentation.

4.3.2 Classification of approaches

In the following, approaches to synchronization in systems with cooperative data management are classified. The classes mainly differ in the degree of cooperation they support and in the power and flexibility of the synchronization scheme. The selection of a specific synchronization model implies the use of a conforming recovery method.

4.3.2.1 Conservative approaches

In most early CAD-databases, the concept of a checkout-modify-checkin cycle was used to manipulate complex data-objects. By executing checkout, a user can reserve a data-object for exclusive access. After changing the object, the user makes his modifications available to others by executing checkin. In this approach, some data may be unavailable to all but one user for a long time, which limits cooperation considerably. Examples can be found in [HHMM88], [KLMP84], [LoPl83], [LeHo88].

4.3.2.2 Floor control

Many applications which allow editing of an object by several participants in a cooperative session, introduce the notion of a floor. The floor holder is a single user who can execute operations of the application at each point of time, all other users can only watch his actions. To allow real cooperation, the floor can be passed between the participants according to a special floor passing protocol. The degree of cooperation in this approach can be increased, by reducing the size of the objects for which a floor exists, but most systems use one floor per application. Systems using floor control are described in [CMB*90], [KnPr90], [AGN88], [GrSa87].

4.3.2.3 Advanced transaction concepts

To support the requirements imposed on data management by long-lasting and cooperative transactions on complex data-objects, new transaction concepts have been developed. Most of the new concepts are based on hierarchically structured transactions like open nested transactions. The structure of a transaction reflects either the structure of the problem it has to solve, or the organizational structure of the group of people, who cooperate in the transaction. To enhance data availability, a subtransaction can commit, before its parent commits. To abort a parent transaction, compensating transactions are needed for committed subtransactions. A wide field of cooperation schemes can be realized by defining application-specific rules for synchronization and recovery. Transaction models supporting cooperation are described in [BKK85], [HOS90], [NRZ92], [HHZ*92], [MRKN92].

4.3.2.4 Flexible locking mechanisms

As the locking of data against access by other users is somehow contradicting the idea of cooperation, locking concepts with more flexible semantics have been developed. In the following, some examples for these concepts are briefly described. A tickle lock is a lock which will be automatically released, if the lock-holder does not access the locked data for some time, or if someone else wants to access the data ([GSW86], [GrSa87]). A notify lock will inform the holder of a read lock, if someone else wants to write the locked data ([ReKa83], [GSM86], [TSH86]). In a probabilistic lock mechanism, locks will be established, if no conflicts occur ([GrSa87]). Otherwise the user or application trying to get a lock will be informed and can decide what action to take. Another method of flexible synchronization is to allow several users to create different versions of a data item. Versions of the same data item have to be merged later. To resolve conflicts between different versions, user interaction may be necessary ([GrSa87], [NRZ92]).

4.3.2.5 Social protocols

Another approach to synchronization in cooperative systems is to allow total user freedom. In this case, users have to synchronize themselves by means outside the application, so-called social protocols. The computer system might or might not provide a channel over which a social protocol can be run. Systems using social protocols are described in [EGR91], [KLMW91], [FoSt86], [SFB*87].

All these concepts may appear on their own, but in general a system will adopt a combination of them.

4.3.3 Existing systems

In this section, some examples from the literature are discussed, which allow cooperation of users or applications on shared multimedia data. The integration of continuous media is very rarely discussed in the literature on cooperation.

4.3.3.1 Mesa file system

The Mesa file system was developed at XEROX PARC to allow cooperation between different processes ([ReKa83]). The interface of the file system contains functions for cooperative lock management. The locks used in Mesa are similar to the already mentioned notify locks. A process acquires a lock by defining the lock mode and a callback function 'PleaseReleaseProc' that will be called by the file system, when another process acquires a conflicting lock. A request to release a lock may be accepted, rejected, or deferred. A process which cannot acquire a lock instantly, may supply to the file system a callback function 'NotifyProc', which will be called, as soon as the previously acquired lock can be granted.

The Mesa file system was the only approach found in the literature, which includes cooperation into a file system. In most other approaches, cooperation is handled by a DBMS or by a specialized application which itself uses a conventional DBMS or file system.

Mesa enhances availability of shared data and allows relatively fast propagation of changes. Correctness, minimization of lost work, partial undo and multilevel synchronization have to be realized in applications based on Mesa. As stated in [ReKa83], writing those applications is not an easy task.

4.3.3.2 CoDraW

CoDraW is a sketching tool, that can manipulate a common drawing workspace containing graphical and textual objects ([KLMW91]). The main focus of CoDraW was to achieve as much freedom for a single user's actions as possible. CoDraW is an example for a cooperative application, where all participants run an instance of the application which has replicated all data objects that are manipulated in the session. To synchronize the instances of CoDraW in a session, the operations executed by each participant are broadcast to all instances. To ensure data integrity, a participant must hold a lock on an object, before manipulating it. In CoDraW, locks are established, by communicating with all other participants. If another participant holds a conflicting lock, one or both users are notified of the conflict and may take appropriate actions, perhaps using a social protocol.

In CoDraW, data are highly available to all users, because of full data replication and notify locks. This is qualified by the fact, that communication must be possible with all participants, to establish locks. Changes to data are propagated very fast to all participants, even the 'rubber-banding' of some drawing operations is displayed on all screens. As all data synchronization is done by a specialized application, this application ensures correctness criteria, lost work is minimized by user intervention, partial undo is not yet addressed.

4.3.3.3 Collaborative Editing System (CES)

CES ([GSW86], [GrSa87]) supports a group of users in writing a structured text document, its concepts could nevertheless be used for multimedia documents as well. A document in CES consists of an outline, which is replicated on all participant's machines, and a set of sections containing text. Each section is stored on the machine of its original creator. CES was implemented in Argus, a programming language including transaction concepts to manage persistent and atomic data types. Users of CES can display parts of the outline and of one or more sections. A user can be editing one section at a time, for which he has a tickle lock. CES can commit every few keystrokes of an editing user, so that other users see a nearly up-to-date version of a section being edited. New committed versions of a section are put on a version stack, so that a user can undo his changes step by step. By executing a checkpoint operation, older versions can be removed from the stack. To implement CES, the concept of an active transaction would have been useful, as for example problems were encountered when screens of other users had to be updated as a consequence of a transaction abort. For this purpose, user defined actions bound to begin or end of transactions are desirable, but were not available in Argus.

In CES, data availability is raised by the use of tickle locks, changes are propagated rather promptly by commit of partial actions. Application specific correctness criteria, together with minimal lost work and partial undo are achieved by the transaction mechanism using tickle locks and version stacks. Data are primarily synchronized on the level of persistent stored data, which induces problems when trying to synchronize screen images.

4.3.3.4 Cooperative Transaction Model

At Brown University, a cooperative transaction model for design databases was developed ([NRZ92]). In this model, a top-level transaction containing all design activities for some object, consists of a hierarchy of transaction groups (TG) and cooperative transactions (CT), where the latter constitute the leaves of the transaction tree. Figure 7 shows an example of a transaction hierarchy realizing the design of a car. Each transaction group represents some part of the design team, involved in a common task. Members of a transaction group may be other transaction groups or cooperative transactions. A cooperative transaction is a collection of atomic operations executed by a single designer. Each TG defines correctness criteria for the operations issued by its members. Patterns define sequences of operations which must occur in a

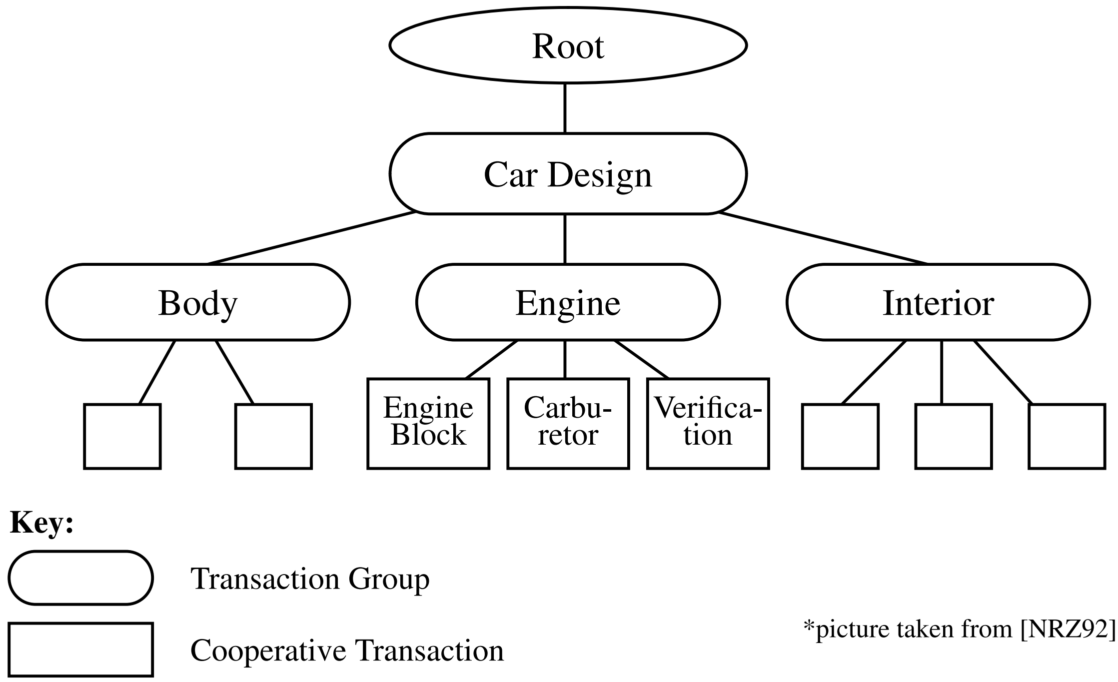

Figure 7: Example for a hierarchical design transaction

correct execution, whereas conflicts define illegal sequences of operations. These patterns and conflicts may reflect the style of work of a design team, legal constraints to the design process etc. and differ from one TG to another. A TG has a private copy of each object which is accessed by one of its members. All member operations affect the copy of the TG. By performing a checkpoint, a TG may merge its version with the version of its parent TG. To do this, a TG will supply an equivalent summary of the operations that were executed on its local version, to its parent. The parent TG will check those checkpointed operations against its own patterns and conflicts.

To allow recovery, a log has to be kept in each TG, that contains information about all committed operations and the patterns and conflicts, they are involved in. If an operation has to be undone, operations of the same or other CTs may become invalid according to patterns and conflicts. For each recovery action, patterns and conflicts have to be rechecked, and humans may be involved in the decision, which operations really must be undone.

Data availability is enhanced, because TGs keep local versions and because only the relatively short operations are atomic. Between the CTs of a TG, data can be propagated relatively fast, but no automatic notification of changes is done. The primary goals of the model include realization of application specific correctness criteria, minimal lost work and partial undo. Synchronization is only done on the database level.

In the context of a software engineering environment, a similar transaction model has been proposed at Brown University, which adds the notion of active transactions, triggering user defined actions on begin, commit or abort of transaction ([HHZ*92]). Notification of data changes can by achieved in this model.

4.3.3.5 Publication environment of GMD-IPSI

At GMD-IPSI, an open publication environment was proposed for multimedia documents ([MRKN92]). Multimedia data are modelled as objects, each object class having a fixed set of operations, being executed by software components, called *tools*. Tools are independent from the publication environment and may have existed before the latter. To add new object classes to the environment, the appropriate tools have to be provided too. The principal components of the environment are shown in figure 8. A user accesses the environment through a so-called *as-*

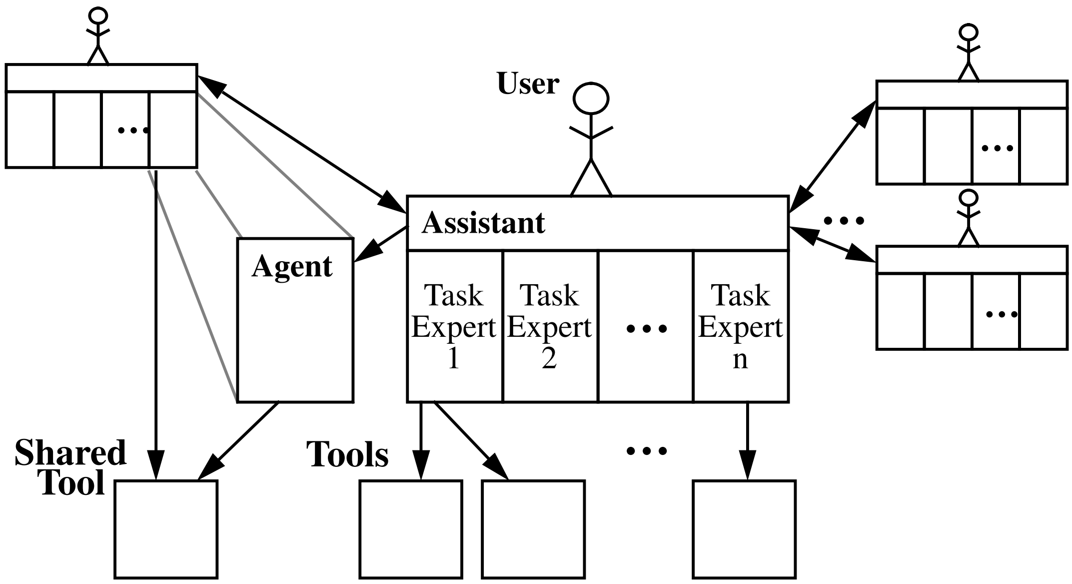

Figure 8: Application structure of the publication environment

sistant. A user's assistant contains one or more *task experts* which each can utilize a set of tools. The assistants of different users differ according to the role, the user plays in the publication process. If a user wants some task to be done, it requires this task from its assistant. If the assistant has an appropriate task expert, this will access the right tool. Otherwise the assistant may ask an assistant of some other user. This assistant may know how to do the task, it may ask another assistant or its own user. Thereby, users of different knowledge and in different roles can cooperate via their assistants. An assistant may generate a software component called *agent*, which has the basic functionality of one of the assistant's task experts. An agent can be exported to another user, whose assistant then can access the corresponding tool more directly.

The interaction between assistants, task experts, agents and tools is structured according to the transaction concept of open nested transactions ([WeSc92]), which is a generalization of multilevel transactions, introduced by Weikum and others in [WeHa91]. In the publication environment, each operation of one software component on another is modelled as a subtransaction. On any level of the transaction tree, application specific correctness criteria can be specified, by defining conditions under which two operations are commutative. This can be done on the object level, as each object class has a fixed set of operations.

High data availability and application specific correctness criteria are achieved by defining commutativity between operation. As compensating operations must exist for all operations, lost work can be minimized and partial undo is possible. Changes to data are not automatically notified to other users. Synchronization is only done on the persistent data by the transaction mechanism.

4.3.4 Discussion

To conclude Section 4.3, the following remarks give a short summary of data synchronization concepts in existing cooperative systems and identify future research topics.

In the database context some approaches exist that allow user cooperation, but as in this field data integrity is a major claim, data synchronization is done on persistent data. This involves rather costly and time-consuming operations, thereby restricting cooperation.

On the other hand, a lot of cooperative applications have been developed in the research context of CSCW. But, as this is a quite new field, most applications have a somehow ad-hoc character. Two main concepts can be identified:

- Application and data are fully replicated, the replicas are brought up-to-date by broadcasting the user input. Synchronization is done on the applications' workspaces by a broad spectrum of mechanisms from floor control to chaotic interaction, controlled by social protocols.
- The application and data are centralized, broadcasting output to all participants. In this case, user inputs are normally synchronized by a floor passing mechanism, data synchronization is done on the displayed data.

Combinations of these two approaches are found as well, but the general concepts for data synchronization are not yet identified.

5 Conclusion

Document architectures differ in their scope because usually they have been designed with different applications in mind. The integration of time dependent information types requires extensions with respect to structuring and specification of temporal relationship. The extensions will have to be adapted to specific structuring schemes of the different document architectures.

Apart from specification issues, the introduction of the dimension time will influence considerably the way documents are created, stored and presented.

In Section 2.4 and in chapter 3 we have presented different aspects of synchronization in the context of multimedia documents. Two levels of synchronization were distinguished: document (high-) level and stream (low-) level synchronization.

The former refers to the core issues which make up a multimedia document: presentation structure and user interference. Several schemes were presented and assessed for their deficiencies. It was found necessary, that presentation specification schemes allow for incorporating actions with non-deterministic durations. Not doing so, excludes actions representing user behaviour (which inherently are non-deterministic). The specification power of time system based schemes was found unsatisfactory. Expanding it by the concepts of action driven schemes is viewed as a worthwhile consideration. Finally, a concept for presentation style control, based on separating object and global time was presented. It was found appropriate in allowing for specifying adequate presentation control functions.

Synchronization at the stream level already guarantees user-perceivable end-to-end qualities, such as bounded delay, jitter and skew. Providing for these relies on a set of mechanisms. First, end-to-end connections have to be considered, since end-to-end qualities are expected. In contrast to transport system connections, multimedia end-to-end connections span all resources between the source and the sink of a stream (including camera devices, compressors, transport systems, etc.). The synchronization functions relate to these connections. They were classified as delay, jitter and skew control, start-up synchronization and synchronization event handling. Approaches to each of them were presented. Many of these are well-understood and will probably become stable solution approaches.

The above separation immediately gives rise to the question, how the presented low-level mechanisms can be employed to enforce document level specifications. It turns out, that these questions have been rarely addressed yet. A first step in this direction was discussed as (the low-level) synchronization event generation. It is clearly a support means for enforcing presentation scheduling. Yet, further steps taking into account delays due to distribution, non-deterministic user behaviour, etc. have to follow. Presentation style control is also an issue which still has to be investigated thoroughly. Stream connections as discussed in Section 3 are usually set up with QoS parameters which are not supposed to change during their lifetime. However, dynamic control exerted by a user, changes these values during connection lifetime. This dynamic adaption is currently not supported at the level of stream synchronization.

All in all, bridging the gap between document level and stream level synchronization, is still an open research issue, which will require considerable research effort.

In Section 4 we presented three different aspects of data management for multimedia data: Concepts for data modelling and storage on two different levels of abstraction, namely DBMS and storage system technology; protocols needed to access multimedia data on remote systems; and support for user cooperation on shared multimedia data.

Database support for multimedia documents requires additional facilities to those provided in current relational database systems. These comprise

- support for multimedia data types in both the query language and the description of records,
- support for the representation of structural and relationship information between parts of a document,
- stream-oriented data transfer mechanisms suitable for the piecewise access to large data units and the real-time access of continuous media data.

Although it is partly possible to extend relational database technology to satisfy the needs of multimedia document storage, object-oriented database systems are more suitable as a base for a MMDBMS, due to better support for the representation of the needed information types and the extensibility inherently available in object-oriented systems.

Todays digital storage systems are able to store all types of data in multimedia documents, with the exception of uncompressed video data. The existing file systems do not support the storage of digital audio and video data, because they cannot guarantee the real-time requirements for storing and retrieving continuous media. New concepts for file systems, extending them to multimedia file systems, are needed to store digital audio and video data.

Remote access to MMDBMS based on the access protocols RDA or DFR suffers from the following problems:

- the request-response technique used by both protocols allows only the access to data units as a whole, which is not suitable for continuous media data or even large discrete data types,
- real-time transfer of continuous media data is neither supported by these access protocols themselves nor by the supporting lower layer protocols.

Extending RDA or DFR to remove these limitations requires major changes to both the underlying design concepts and the concrete specifications of these protocols. The lower layer protocols supporting DFR and RDA require extensions for the isochronous transfer of data too.

Concerning user cooperation on multimedia documents, the requirements imposed on data management have been identified. Some existing approaches to cooperative applications were evaluated according to those requirements. Resulting from this examination, the following issues can be identified for future research:

- As both, integration of multimedia data in documents and multiuser cooperation on documents, are rather new research fields, the requirements to data management systems are not yet fully understood for the case, where both topics come together.
- As most existing applications are rather isolated solutions, general data management concepts have to be defined, that take into account the requirements of highly cooperative applications, where synchronization of data is concerned. In particular, a general architecture, supporting synchronization on all three data-levels, persistent data, application workspace and display data is lacking.

6 Bibliography

[ACA91] N. Almeida, J. Cabral, and A. Alves. End-to-end Synchronization in PAcket Switched Networks. *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[AGH90] D. P. Anderson, R. Govindan, and G. Homsy. Abstractions for Continuous Media in a Network Window System. *Report No. UCB/CSD 90/596, EECS, Univ. of California, Berkeley*, 9 1990.

[AGN88] Hussein M. Abdel-Wahab, Sheng-Uei Guan, and Jay Nievergelt. Shared Workspaces for Group Collaboration: An Experiment Using Internet and UNIX Interprocess Communications. *IEEE Communications Magazine*, pages 10–16, 11 1988.

[AHS90] D. P. Anderson, R. G. Herrtwich, and C. Schaefer. SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet. *Report No. UCB/CSD 90/562, EECS, Univ. of California, Berkeley*, 2 1990.

[Ande90] D. P. Anderson. Meta-schedulig for Distributed Continuous Media. *Report No. UCB/CSD 90/599, EECS, Univ. of California, Berkeley*, 10 1990.

[AnFe88] D. P. Anderson and D. Ferrari. The DASH Project: An overview. *Report No. UCB/CSD 88/405, EECS, Univ. of California, Berkeley*, 2 1988.

[AnHe90] D. P. Anderson and R. G. Herrtwich. Resource Management for Digital Audio and Video. 1 1990.

[AnHo91a] D. P. Anderson and G. Homsy. A Continuous Media I/O Server and Its Synchronization Mechanism. *IEEE Computer*, 10 1991.

[AnHo91b] D. P. Anderson and G. Homsy. Synchronization Policies and Mechanisms in a Continuous Media I/O Server. *Report No. UCB/CSD 91/617, EECS, Univ. of California, Berkeley*, 2 1991.

[AOG91] David P. Anderson, Yoshitomo Osawa, and Ramesh Govindan. Real-Time Disk Storage and Retrieval of Digital Audio/Video Data. *UC Berkley TR UCB/CSD 91/646*, 9 1991.

[ATW*89] D. P. Anderson, S.-Y. Tzou, R. Wahbe, R. Govindan, and M. Andrews. Support for Continuous Media in the DASH System. *Report No. UCB/CSD 89/537, EECS, Univ. of California, Berkeley*, 10 1989.

[BaMa91] A. Banerjea and B. A. Mah. The Real-Time Channel Administration Protocol. *2nd Intl Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[BHL91] G. Blakowski, J. Huebel, and U. Langrehr. Tools for Specifying and Executing Synchronized Multimedia Presentations. *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[BHSS90] F. Ball, D. Hutchson, A. Scott, and D. Shepherd. A Multimedia Network Interface. *3rd IEEE COMSOC Intl. Multimedia Workshop, Bordeaux, France*, 11 1990.

[BKK85] F. Bancilhon, W. Kim, and H. Korth. A Model of CAD Transactions. In

Proceedings of the 11th International Conference on Very Large Databases, pages 25–33, Stockholm, Sweden, 1985.

[CGHS91] G. Coulson, F. Garcia, D. Hutchison, and D. Shepherd. Protocol Support for Distributed Multimedia Applications. *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[CMB*90] T. Crowly, P. Milazzo, E. Baker, H. Forsdick, and R. Tomlinson. MMConf: An Infrastructure for Building Shared Multimedia Applications. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 329–342, New York, 10 1990. ACM SIGCHI & SIGOIS, ACM Press.

[CTH*86] S. Christodoulakis, M. Theodoridou, F. Ho, M. Papa, and A. Pathria. Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System. *ACM Trans. on Office Inf. Syst.*, 4(4), 10 1986.

[DFR91] ISO International Standard 10166: Information Technology - Document Filing and Retrieval (DFR), 1991.

[EGLT76] K.P. Eswaran, J.N. Gray, R.A. Lorie, and I.L. Traiger. The Notions of Consistency and Predicate Locks in a Database System. *Communications of the ACM*, 19(11):624–633, 11 1976.

[EGR91] C.A. Ellis, S.J. Gibbs, and G.L. Rein. Groupware: Some Issues and Experiences. *Communications of the ACM*, 34(1):38–58, 1 1991.

[Elma92] Ahmed K. Elmagarmid, editor. *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, San Mateo, California, 1992.

[Ferr91] D. Ferrari. Design and Applications of A Delay Jitter Control Scheme for Packet-Switching Internetworks. *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[FeVe90] D. Ferrari and D. C. Verma. A Scheme for Real-Time Channel Establishement in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3), 4 1990.

[FoSt86] G. Foster and M. Stefik. Cognoter, theory and practice of a Colab-orative tool. In *Proceedings of the Conference on Computer Supported Cooperative Work '86*, pages 7–15, 1986.

[Gall91] Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, 4 1991.

[GBD*91] S. Gibbs, C. Breiteneder, L. Dami, V. de Mey, and D. Tsichritzis. A Programming Environment for Multimedia Applications. *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[GDT91] S. Gibbs, L. Dami, and D. Tsichritzis. An Object-Oriented Framework for Multimedia Composition and Synchronistion. *Eurographics Multimedia Workshop, Stockholm*, 1991.

[Gibb91] S. Gibbs. Composite Multimedia and Active Objects. *Proc. OOPSLA'91*, pages 97–112, 1991.

[GoAn91] R. Govindan and D. P. Anderson. Scheduling and IPC Mechanisms for Continuous Media. *Report No. UCB/CSD 91/622, EECS, Univ. of California,* *Berkeley*, 4 1991.

[Gold90] Charles F. Goldfarb. *The SGML HANDBOOK*. Oxford University Press, New York, 1990.

[Gold91] Charles F. Goldfarb. HyTime: A Standard for structured hypermedia interchange. *Computer*, 8 1991.

[Gray78] J.N. Gray. Notes on Data Base Operating Systems. In *Operating Systems*, volume 60 of *Lecture Notes in Computer Science*, pages 394–481. Springer Verlag, 1978.

[GrSa87] Irene Greif and Sunil Sarin. Data Sharing in Group Work. *ACM Transactions on Office Information Systems*, 5(2):187–211, 4 1987.

[GSM86] L.N. Garret, K.E. Smith, and N. Meyrowitz. Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System. In *Proceedings of the Conference on Computer Supported Cooperative Work '86*, pages 163–174. ACM, 1986.

[GSW86] I. Greif, R. Seliger, and W. Weihl. Atomic Data Abstractions in a Distributed Collaborative Editing System. In *Proceedings of the 13th Annual Symposium on Principles of Programming Languages*, pages 160–172, New York, 1 1986. ACM.

[Haas88] L.M. Haas. Supporting Multi-Media Object Management in a Relational Database Management System. Technical report, IBM Almaden research Center, 1988.

[HaLo82] R. Haskin and R. Lorie. Using a relational Database Systems for Circuit Design. *IEEE Database Engineering*, 5(2):10–14, June 1982.

[Herr92] R. G. Herrtwich. An Architecture for Multimedia Data Stream Handling and Its Implication for Multimedia Transport Service Interfaces. *IEEE ?*, 1992.

[HGA90] G. Homsy, R. Govindan, and D. P. Anderson. Implementation Issues for a Network Continuous-Media I/O Server. *Report No. UCB/CSD 90/597, EECS, Univ. of California, Berkeley*, 9 1990.

[HHMM88] T. Haerder, Ch. Huebel, K. Meyer-Wegener, and B. Mitschang. Processing and transaction concepts for cooperation of engineering workstations and a database server. *Data and Knowledge Engineering*, 3:87–107, 1988.

[HHS91] D. Hehmann, R. G. Herrtwich, and R. Steinmetz. Creating HeiTS: Objectives of the Heidelberg High-speed Transport System. *F&E-Projektberichte, GI-Jahrestagung 1991, Darmstadt*, 10 1991.

[HHZ*92] Sandra Heiler, Sara Haradhvala, Stanley Zdonik, Barbara Blaustein, and Arnon Rosenthal. *A Flexible Framework for Transaction Management in Engineering Environments*, chapter 4, pages 87–121. In Elmagarmid [Elma92], 1992.

[HKL*91] Kevin Harney, Mike Keith, Gary Lavelle, Lawrence D. Ryan, and Daniel J. Stark. The i750 Video Processor: A Total Multimedia Solution. *Communications of the ACM*, 34(4):64–78, 4 1991.

[HKN89] R. Hunter, P. Kaijser, and F. Nielsen. ODA: a document architecture for open systems. *Computer Communications*, 4 1989.

[HNV92] R. G. Herrtwich, R. Nagarajan, and C. Vogt. Guaranteed-Performance Multimedia Communication Using ST-II Over Token Ring. *submitted to the Intl. Conference on Distributed Computer Systems (ICDCS 92')*, 1992.

[HODA92] HyperODA - Extensions for Temporal Relationships, 1992. ISO/IEC JTC 1/SC 18/WG 3/N2232.

[Hoep91a] P. Hoepner. Presentation Scheduling of Multimedia Objects and Its Impact on Network and Operating System Support. *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[Hoep91b] P. Hoepner. Synchronisation der Praesentation von Multimedia-Objekten - Modell und Beispiele. *Informatik-Fachberichte 293: Telekommunikation und multimediale Anwendungen der Informatik*, pages 455–464, 10 1991.

[HOS90] W.H. Harrison, H. Ossher, and P.F. Sweeney. Coordinating Concurrent Development. In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 157–169. ACM, 10 1990.

[HSS90] Hehmann, Salmony, and Stuettgen. Transport Services for Multimedia Applications on Broadband Networks. *Computer Communications*, 13(4), 1990.

[HYTI92] Information Technology - Hypermedia/Time-based Structuring Language (HyTime), 1992. ISO/IEC 10744.

[IMS92] PHILIPS IMS, editor. *Introducing CD-I*. Addison-Wesley, 1992.

[KLMP84] W. Kim, R. Lorie, D. McNabb, and W. Plouffe. A transaction mechanism for engineering design databases. In U. Dayal, G. Schlageter, and L.H. Seng, editors, *Proceedings of the International Conference on Very Large Databases*, pages 355–362, 1984.

[KLMW91] Thomas Kirsche, Horst Lührsen, Klaus Meyer-Wegener, and Hartmut Wedekind. A distributed sketching tool featuring freedom of action and equality of all users. 1991?

[KnPr90] M.J. Knister and A. Prakash. DistEdit: A distributed toolkit for supporting multiple group editors. In *Proceedings of the 3rd Conference on Computer Supported Cooperative Work*, pages 343–355. ACM, 1990.

[KST92] W. Kim, M. Scheevel, and C. Tomlinson. Object-oriented databases for new applications. *Future generation Computer Systems*, 7:317–327, 1991/1992.

[LeHo88] B.T. Lewis and J.D. Hodges. Shared Books: Collaborative publication management for an office information system. In *Proceedings of the Conference on Office Information Systems*, ACM SIGOIS Bulletin, pages 197–204, 1988.

[LiGh90a] T. D. C. Little and A. Ghafoor. Network Considerations for Distributed Multimedia Object Composition and Communication. *IEEE Network Magazine*, 11 1990.

[LiGh90b] T. D. C. Little and A. Ghafoor. Synchronization and Storage Models for Multomedia Objects. *IEEE Journal on Selected Areas in Communications*, 8(3), 3 1990.

[LiGh91] T. D. C. Little and A. Ghafoor. Scheduling of Bandwidth-Constrained Multimedia Traffic. *2nd Intl. Workshop on Network and Operating System*

Support for Digital Audio and Video, 11 1991.

[LiLa73] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1), 1 1973.

[LoPl83] R. Lorie and W. Plouffe. Complex objects and their use in design transactions. In *Proceedings Data Base Week: Engineering Design Applications*, pages 115–121, 1983.

[Luth91] Arch. C. Luther. *Digital Video in the PC environment*. McGraw-Hill, 2nd edition, 1991.

[Meiß91] Klaus Meißner. Architectural Aspects of Multimedia CD-I Integration in UNIX / X-Windows Workstations. In *2nd International Workshop on Network and Operating System Support for Digital Audio and Video*, Heidelberg, 11 1991.

[Meye91] Klaus Meyer-Wegener. *Multimedia Datenbanken*. Leitfäden der angewandten Informatik. Teubner Stuttgart, 1991.

[MHEG92] Information processing - Coded Representation of Multimedia and Hypermedia Information Objects (Working Draft), 1992. JTC1/SC29/WG12.

[Moss81] J. Eliot B. Moss. Nested Transactions: An Approach to Reliable Distributed Computing. Tech. Report 260, M.I.T. Laboratory for Computer Science, 4 1981.

[MRKN92] Peter Muth, Thomas C. Rakow, Wolfgang Klas, and Erich J. Neuhold. *A Transaction Model for an Open Publication Environment*, chapter 6, pages 159–218. In Elmagarmid [Elma92], 1992.

[NKN91] Steven R. Newcomb, Neill A. Kipp, and Victoria T. Newcomb. The Hypermedia/ Time-based Document Structuring Language "HyTime". *Communications of the ACM*, 11 1991.

[NRZ92] Marian H. Nodine, Sridhar Ramaswamy, and Stanley B. Zdonik. *A Cooperative Transaction Model for Design Databases*, chapter 3, pages 53–85. In Elmagarmid [Elma92], 1992.

[ODA89] Information processing - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format, 1989. ISO/IEC 8613.

[RaVi91] P. Venkat Rangan and Harrick M. Vin. Designing File Systems for Digital Video and Audio. *Procceedings of the 13th ACM Symposium on Operating Systems Principles in ACM Operating Systems Review*, 25(5):81–94, 10 1991.

[RDAG90] ISO Draft Proposal 9579-1: Remote Database Access, Part 1: Generic Model, Service and Protocol, 1990.

[ReKa83] L. G. Reid and P.L. Karlton. A File System Supporting Cooperation Between Programs. In *Proceedings of the 9th Symposium on Operating Systems Principles*, pages 20–29, New York, 1983. ACM.

[Ripl89] G. David Ripley. DVI - a digital multimedia technology. *Communications of the ACM*, 32(7):811–822, 7 1989.

[ROSE89] ISO International Standard 9072: Information Processing Systems - Text Communication - Remote Operations, 1989.

[SFB*87] M. Stefik, G. Forster, D. Bobrow, K. Kahn, S. Lanning, and L. Suchman. Beyond the Chalkboard - Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM*, 30(1):32–47, 1 1987.

[SGML86] Information processing - Text and office systems - - Standard Generalized Markup Language (SGML), 1986. ISO/IEC 8879.

[Sijs91] F. Sijstermans. CD-I Full-Motion Video encoding on a parallel Computer. *Communications of the ACM*, 34(4):81–91, 4 1991.

[Smit89] Joan M. Smith. Standard Generalized Markup Language and related standards. *Computer Communications*, 12(2), 4 1989.

[SRR90] R. Steinmetz, J. Rückert, and W. Racke. Multimedia-Systeme. *Informatik Spektrum*, 13(5):280–282, 1990.

[StEd91] C. V. Steffen and T. H. Edgar. Image and video coding standards and their implementations. In *1st Australien Multi-Media Communications, Applications & Technology Workshop*, pages 205–222, 1991.

[SW*90] A. Scheller, E. Wilde, et al. Definition of an Audio Architecture including a proposal for an ODA Audio Content Architecture. BERKOM-Project:"Multi-Media-Dokumente im ISDN-B" der Gesellschaft für Mathematik und Datenverarbeitung, Forschungszentrum für offene Kommunikationssystem, 1990.

[TeSw88] Douglas B. Terry and Daniel C. Swinehart. Managing Stored Voice in the Etherphone System. *ACM Transactions on Computer Systems*, 6(1):3–27, 2 1988.

[TSH86] R.H. Trigg, L.A. Suchman, and F.G. Halasz. Suporting collaboration in NoteCards. In *Proceedings of the Conference on Computer Supported Cooperative Work '86*, pages 153–162. ACM, 1986.

[WeHa91] Gerhard Weikum and Christof Hasse. Multi-Level Transaction Management for Complex Objects: Implementation, Performance, Parallelism. Technical Report 162, Department of Computer Science, ETH Zürich, 1991.

[WeSc92] Gerhard Weikum and Hans-J. Schek. *Concepts and Applications of Multilevel Transactions and Open Nested Transactions*, chapter 13, pages 515–553. In Elmagarmid [Elma92], 1992.