Multimedia Authoring Tools:
State of the Art and Research Challenges

Dick C.A. Bulterman and Lynda Hardman

CWI: Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam
E-Mail: {Dick.Bulterman, Lynda.Hardman}@cwi.nl

Abstract: The integration of audio, video, graphics and text on the desktop promises to fundamentally challenge the centuries-old model of the printed document as the basis for information exchange. Before this potential can be realized, however, systems must be devised that enable the production and presentation of complex, inter-related media objects. These systems are generically called multimedia authoring tools. In this article, we consider the development of multimedia authoring tools, examine the current state of the art, and then discuss a set of research challenges that need to be addressed before the full potential of multimedia output technology can be effectively utilized to share information.

1 Introduction

The encoding and distribution of information has been a problem that has kept much of mankind busy since its beginnings. Perhaps the most significant milestone in this activity was the development of the written word, which allowed spoken communication to be reliably reproduced and distributed under a measure of control of its originator. The technology available for producing written documents remained relatively constant until a few hundred years ago, when significant refinements in the means for creating and distributing documents began to take place. Fig. 1 summarizes some of these refinements, including the printing press, the postal service, the development of newspapers (the catalyst for current wide-scale literacy), the xerographic copier and, more recently, the combination of computer-based page composition systems with the com-

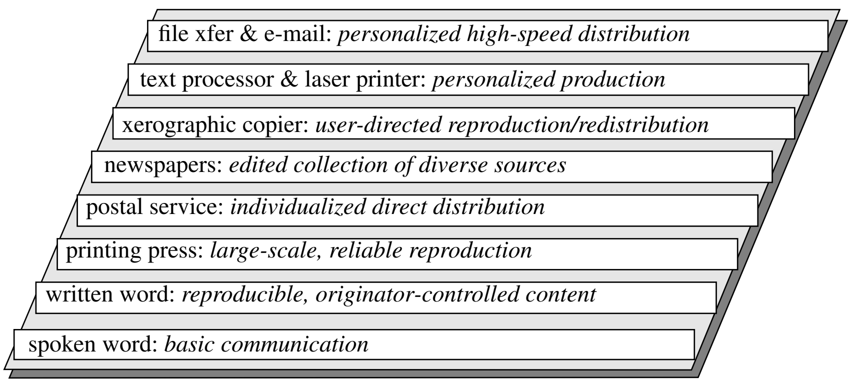


Figure 1. Significant developments in text processing and distribution.

puter network and laser printer. Each of these developments addressed specific user needs to make text-based information processing more effective, reliable, reproducible and broadly-available.

During the past five years, an enormous increase in computer processing speed and a dramatic reduction in the cost of intelligent peripheral devices have created the potential to augment and even replace text as the standard form of information exchange. These developments have been grouped under the generic name of *multimedia*. Unlike the user-driven development of text processing facilities, much of the growth of multimedia has resulted from the relatively independent development of the underlying output technology. This has led to a gap between a computer's ability to process and present information and the user's ability to create and manage documents that exploit the power of the technology available.

The problem of creating and managing multimedia information is not trivial. Consider the introduction of audio to the desktop: in spite of a lifetime of experience in producing and processing spoken information, 'audio e-mail' has not developed as a natural alternative to text-based electronic mail. The major drawback is not the lack of familiarity with the medium, but rather the lack of integrated authoring and editing tools that enable a broad class of users to access the technology and to create messages that can be easily shared with others. The creation and editing problems include the ability to visualize, rearrange and refine documents. These problems are compounded when several media must be manipulated together and when media are used that require specialized training before they can be used for general information exchange.

In the following paragraphs, we review the development of applications that support the construction and presentation of multimedia documents. Section 2 reviews the building blocks used by authoring systems, including media object creation and document distribution technologies. Section 3 discusses the current state of the art for supporting multimedia authoring, including authoring paradigms and an extended example. Section 4 concludes with a discussion of topics that need to be addressed in the development of next-generation authoring systems.

2 Media Objects and Multimedia Data Distribution

A *multimedia document* can be viewed as an activity specification that can be used to coordinate the runtime presentation of a collection of *media objects*, where each media object can contain one or more media items, of one or more types. *Multimedia authoring systems* provide a mechanism for defining a multimedia document based on an underlying multimedia information model. The development of multimedia authoring systems encompasses many of the problems and challenges of generalized multimedia data manipulation: data objects must be created, often using special-purpose tools; they must be integrated into a coherent presentation, usually requiring some form of media synchronization; the presentation must be defined in a form that allows distribution to various types of target environments; and the presentation must be 'maintainable' over its commercial or intellectual life-time.

The structure of media objects determines the level of information combination, placement and presentation granularity available to the document author. Consider the abstraction of a document fragment in Fig. 2. Here two choices of object organization are illustrated. On the left side we see a pointer from the document to a composite media object that contains a video sequence, a static illustration, a text overlay and an accompanying soundtrack. The composite object, with dependencies resolved at its source, is fetched and delivered to the processor hosting the presentation. (We ignore for the moment issues of network vs. local access, or encoding in particular formats for particular storage media.) On the right side of the illustration the document references four separate media objects, each of which is individually fetched and sent to the processor. Clearly, the use of composite media objects reduces the burden on the document specification: it only needs to identify the object and provide high-level access control, since the relationships *among* the items are captured within the object definition. The use of a collection of separate object provides extra flexibility—for example, we can choose run-time substitution of objects to tailor the presentation for, say, a multi-lingual audience—at the expense of extra complexity within the document and document presentation system.

2.1 Individual Media Object Definition and Manipulation

Current multimedia data object creation tools are the result of the historical development of a number of separate tools and systems to develop and manipulate separate and distinct types of data. Low-cost graphics adapters have allowed high-resolution bit-mapped displays to become commonplace; low-cost signal processing chips have allowed stereo audio adapters to be integrated into many low- and medium-cost computers; digital cameras and integrated video processing chips have brought full-motion video to the desktop; and inexpensive high-density storage devices such as the CD-ROM have enabled the efficient distribution of the massive amounts of digital data that are generated for and by the above devices.

Where text and drawing systems can usually integrate data creation and editing, data for most other types of media are first created using a capture tool and then refined using an editing tool. Capture and editing are typically done outside the scope of the authoring tool, although some degree of interaction between editing and authoring is often required. Created media objects can be stored as totally separate entities (like

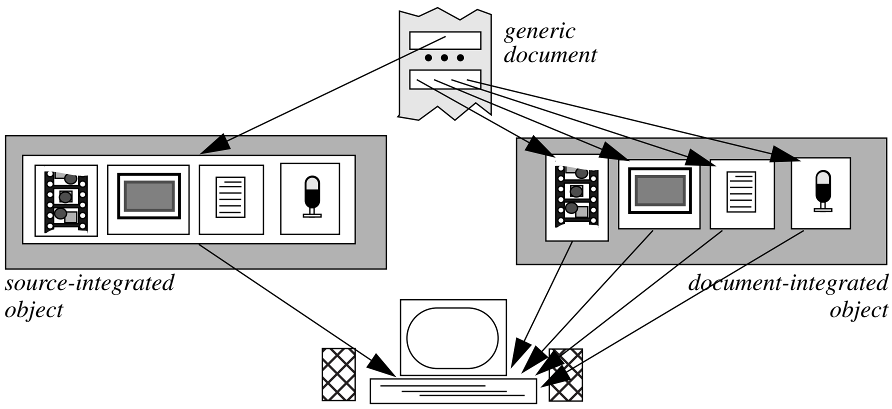


Figure 2. Media objects and documents

user files) or they can be integrated into multimedia databases or file servers. The temporal access constraints of multimedia data usually require special-purpose file- or database-servers to manage storage and delivery of media objects.

Representative capture/editing tools are: [2S] and [2] for video, [9S] and [29] for audio and [1S] and [4S] for image data. A survey of multimedia data servers is given in [15]. A general overview of encoding and delivery aspects of networked multimedia is given in earlier publications in the LNCS series (614, 712 and 846).

2.2 Integrating Separate Objects into a Common Document

Authoring multimedia documents—whether static in structure or dynamically created at runtime—essentially consists of collecting and presenting media objects in a author-desired order, with some degree of navigation control available to the end user. While the degree of composition of each media object will influence the complexity of the authoring/presentation environment, the user-level management of the authoring process remains the most critical element in applying multimedia technology.

Authoring systems are programs that assist the user in managing the creative task of specifying the placement and relative order of media object events. The development of authoring systems can be broadly classified into three generations, each of which are discussed below. Note that the migration of multimedia technology from high-end to low-end systems has had the result that each of the generations are not defined in terms of a time period, but rather in terms of a collected body of facilities available to users at different times on different classes of computing systems.

In identifying breaks in generations, three types of evolutionary change can be identified: changes in the portability/distributability of documents, changes in basic multimedia technology supported within documents and changes in user facilities to access documents. (These parallel the evolutionary forces supporting change in text-based system development.) Performance changes—such as more video frames per second or high audio quality—are not considered to indicate fundamental changes in authoring system functionality.

G_0: First generation authoring systems. The distinguishing characteristics of G_0 include authoring systems that are highly processor-architecture dependent, relying on a high-degree of sophistication of special-purpose graphics and/or audio hardware, with minimal user support for general document browsing. While multimedia is often seen as a new field within computer science, examples of G_0 authoring systems have been present on high-end computing systems and workstations for decades. Representative examples include programs that support air traffic control systems, where real-time graphics and alpha-numeric data are displayed together with static background images, or text formatting systems, where pictures and text integrated into a single print file. More recently, systems like [5S] and [21] provided authoring support that is typical of G_0. Some of these systems provide experimental hypertext navigation support.

G_1: Second generation authoring systems. In G_1, the first evidence of portability of multimedia documents—albeit to a limited degree—is available, often within family members of a particular hardware/software architecture. Support for representation-based document navigation and browsing facilities (that is, navigation/browsing based on the implementation of an object rather than its content) are also included, such as document fast-forward/rewind. In some systems, commercial support for content-based navigation (hyper-links) is also provided, but almost always limited to text data or static images. In G_1 systems, media objects are usually 'leaf nodes': they are treated as self-contained entities with little support for document-level object embedding. The presentation metaphor remains based on a page-by-page model rather than on a single continuous presentation. User input facilities are usually limited to coarse navigation control. Representative examples of G_1 systems are [3S], [10S], and [28].

G_2: Third generation authoring systems. These systems represent the current generation of commercially-available multimedia authoring systems. They have some degree of multi-platform presentation support, enabling basic sharing of documents between Apple Macintosh and IBM PC (compatible) computers. (Support is also available for UNIX workstations, although typically only in research prototype systems.) A limited amount of client-server networking is supported, although most systems rely on local copies of data to provide the required levels of presentation performance. Broad support is available for *de facto* standard output devices, although only rudimentary support is available for emerging standard document and object formats. Representation-based browsing is widely supported, and some systems allow support for content-based searching for dynamic media as well as text/images. Examples and properties of G_2 systems are described in more detail in the next section. In section 4, we consider the needs of characteristics of future generation authoring support.

3 Multimedia Authoring: The State of the Art

This section discusses the major characteristics of current-generation (G_2) authoring systems. We begin with a discussion of development paradigms for authoring and then give an example of the facilities available in an advanced G_2 system.

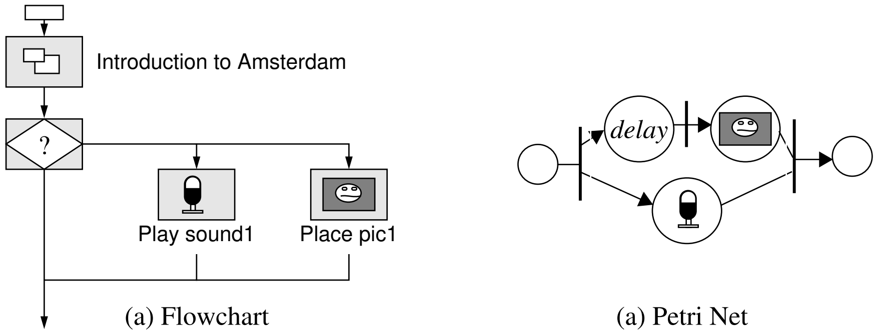
3.1 Approaches To Authoring Multimedia Documents

Four general paradigms exist for multimedia authoring systems:

- *graph-based authoring,* describing a presentation's control flow;
- *timeline-based authoring,* describing a presentation's data flow relative to a common time axis;
- *program-based authoring,* using a text-based specification to describe the positions and timings of individual objects; and
- *structure-based authoring*, which groups objects based on presentation content.

We outline the basic properties and describe representative systems using each paradigm in the following sections. (Note that many authoring environments provide a mixture of paradigms; in our discussion, we classify systems under the dominant paradigm used.)

Graph-based authoring systems. This approach uses a schematic diagram of the control flow interactions among multimedia objects, as illustrated in Fig. 3. Graphs can be informal or formal. Informal graphs are used to give a global illustration of the order-



(a) Flowchart (a) Petri Net

Figure 3. Examples of graph-based authoring paradigms.

ing of relatively coarse object interactions. Formal graphs also illustrate object interactions, but they additionally provide a mechanism for "proving" or determining properties of the presentation based on the semantics of the graph.

Graph-based authoring can provide very powerful presentation descriptions. For all but the simplest documents, however, both informal and formal graphs suffer from a 'complexity explosion' once the size of the presentation grows beyond that found in a simple document. In order to manage this complexity, sophisticated user interface systems are required that allow the user to zoom-in or zoom-out of a presentation. This limits the utility of the approach, especially in the case of informal graphs: here, there is typically no added value of the graph itself aside from an ordering mechanism; once the presentation becomes so complex that it fills multiple physical or screen pages, the integrated view of a presentation is lost. For formal graphs, this added complexity is offset by the results generated from the graph formalism, but even here, the complexity of the description typically limits the use of such models to special-purpose authoring needs such as synchronization analysis.

Graph-based mechanisms also have the disadvantage that detailed interactions among data objects are difficult to describe. For example, in Fig. 3(a), an audio and picture object are shown occurring "at the same time". Suppose that we instead want the audio to start first and then, after a lead-in period, have the picture presented. At one level, the audio and picture are presented in parallel (as shown), but at a more detailed level, the interaction among the items becomes quite complex. Such as relationship can be modelled by the fragment of Fig. 3(b) that is enclosed in the dotted-line box: the bottom node represents the audio fragment, with above it a period of delay followed by the picture. (Note that a similar approach could have been used in the flowchart, at the expense of more detail). In both cases, however, it would be difficult to describe even more detailed relationships, such as: *display the picture when a phrase 'pretty picture' is spoken in the audio object*. It is also difficult to model hyper-navigation behavior in this type of system.

Examples of flowchart-based authoring paradigms are reviewed in [24]. An example of a recent Petri-net-based authoring system is [5]. An example of a hybrid system is Eventor [13], which uses CCS (Calculus of Communicating Systems) as an underlying formal specification mechanism, while providing a flowchart-like user interface. (Eventor also incorporates timeline graphs, as discussed in the following section.)

Timeline-based Authoring Systems. These systems provide a schematic diagram of the dataflow interactions among multimedia objects. A generic timeline is illustrated in Fig. 4. Unlike control flow graphs discussed previously, most of the timeline-based authoring environments use only informal descriptions.

The nature of the timeline is similar to that of a musical score: a number of events are shown in parallel relative to a common axis. This metaphor is especially appropriate to multimedia presentations, with their inherent use of parallel, time-based events. Authoring is carried out by placing icons representing the media items on one of the tracks at a specific time. Items can be stretched across several time frames (either by hand or automatically, depending on the authoring system). Some timeline-based systems allow transition effects (such as fading, blurring, or spatial transposing of objects) to be specified as timeline properties. The granularity of the timeline can be tied to the granularity of the system clock or it can be a user-scalable logical time reference. (Logical clocks ease problems of porting systems, but they do not guarantee that a particular presentation will be able to be presented on all systems.)

The primary advantage of the timeline is that it provides an intuitive ordering of object events. The potential disadvantages with timeline-based authoring systems include the side effects of object editing, run-time control, and presentation navigation. Object editing can be a problem if changes made to individual objects are not correctly integrated back to the presentation timeline. For example, if the start time or duration of a media item is changed (e.g. by editing the data belonging to the item) then the relative positions of other objects on the same or related timelines may also need to change. Such changes are rarely automatic (that is, the 'correct' choice depends on the semantics of the presentation rather than the structure of the timeline); this can require substantial changes in a document. Presentation control involves the imposition of a higher-level control structure on a document (such as a pause facility or the ability to integrate user feedback into the presentation). The control problem of timeline authoring is often solved by introducing a special control track; such a track can be used to define meta-operations such as grouping common presentation elements or indicating when pauses should be inserted in a presentation.

A third problem is presentation navigation. By adjusting a logical time pointer, most timeline systems allow an end-user to jump to specific parts of a presentation. While this type of 'fast-fowarding' or 'fast rewind' in intuitively obvious, the resulting

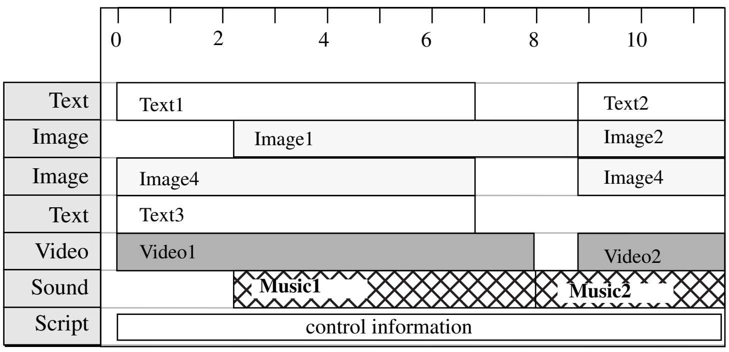


Figure 4. Example of a timeline-based authoring paradigm.

behavior is not: if the time pointer in Fig. 4 is moved to time ‘2’, it is not clear if all objects defined at that time are activated, or only new objects starting at or after the new time. (If one views a video tape as a multi-channel timeline, then an ‘activate all’ model is used. For computer-based presentations, this is much more difficult since each of the composite items need to be independently started and run up to the time indicated by the time pointer.) Navigation problems in timeline result from the use and manipulation of data representations rather than information object contents.

Three example timeline-based systems are Director [8S], Integrator [31] and MAEstro [12]. Of these, perhaps the most widely-used is Director, which was designed for creating animation-based presentations. Graphics, text and video objects can be placed on a timeline, or score as it is termed in the system. The timeline is divided into separate frames, whose speed of playing is determined by the current tempo, where the tempo can be changed at any frame. An object has a position in each frame, and the author can describe a path for the object to follow through a series of frames.

Programming-based authoring systems. Both graph- and timeline-based authoring paradigms make use of illustrations to describe the interaction among media items in a presentation. While this is appropriate for high-level interaction, it is less appropriate for detailed descriptions of complex system behavior. As is illustrated in Fig. 5, a programming-based system gives an author low-level facilities for specifying the components, their timing, layout and interactions within a presentation. The programming effort can range from rapid prototype languages (typically referred to as scripts), through object-based library models, to detailed, low-level extensions to—or replacements of—existing programming languages.

Programming-based approaches can be used to model and manipulate individual media objects or they can be used to coordinate/orchestrate previously created (or dynamic) objects, or both. This has the advantage that rich interaction facilities can often be provided and that at least rudimentary content-based decisions can be made at runtime to guide the presentation of the document. This flexibility comes at a significant price: the definition of the document can be tedious, the portability of a particular document may be limited and the expressive power of the programming interface will be limited by the underlying operating or transport mechanism to provide the control primitives required by the programming interface.

While the programming model appeals to specialists, it is unclear if this model will be accepted by the general multimedia community. Just as there is a trend away from

```
set win=main_win
set cursor=wait
clear win
put background “pastel.pic”
put text “heading1.txt” at 10,0
put picture “gables.pic” at 20,0
put picture “logo.pic” at 40, 10
put text ”contents.txt” at 20,10
set cursor=active
```

Figure 5. Example of a programming-based authoring paradigm.

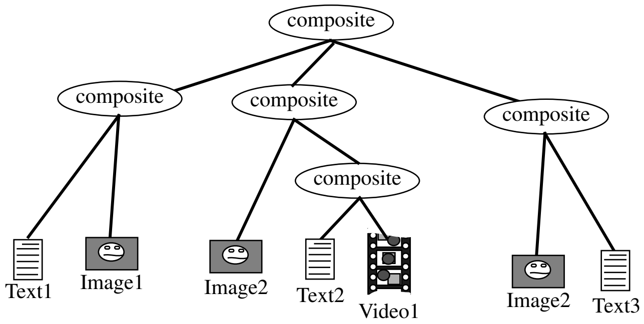


Figure 6. Example of a *structure-based* authoring paradigm.

programming-based text formatters (such as *troff* and *TeX*) to more WYSIWYG text and drawing systems (such as [6S] or [10S]), it may be that programming-based authoring will be reserved for special-purpose documents with critical performance constraints [7S]. Although authoring and object editing systems will be implemented using a hierarchy of programming languages, the user interface to these tools will not be based on the programming paradigm.

Structure-based authoring systems. The three types of authoring paradigms discussed up to this point share a common characteristic: they define documents in terms of the placement and activation of groups of media objects. As is illustrated in Fig. 6, another approach is to separate the definition of the logical structure of a presentation and the media objects associated with a document. In a structure-based document view, the advantages of structured/composite design approaches used in software engineering can be transferred to the multimedia domain. This approach delays the binding of data to documents, allowing an author to develop (and also edit) a presentation outline.

Text-based documents are often developed in terms of a *beginning-middle-end* model, in which the document and often individual sections are constructed in terms of an introduction, the main text body and a summary. Entire documents can then be created in a top-down or bottom-up fashion (or some mixture of the two). Navigation within a document is typically related to its logical structure; detailed reference or content associations can be found locally while more general references are often found further away from a particular point in the document. In multimedia documents, there is usually also a locality of interaction between media objects: synchronization constraints are defined among items that are structurally related more often than items that are farther apart in the document structure.

The definition of a document in terms of an explicit logical structure can be used to partition tasks among several clusters of authors and assist in resource allocation during presentation. The ease with which a document can be edited and maintained also is increased when the author has a good view of the logical relationships between items. Unfortunately, a single structure-based view is typically not sufficient to describe all of the timing and interactions requirements of detailed sections of a presentation. As a result, structure-based authoring is usually combined with other techniques to produce an entire document [20]. We give an example of this in the following section.

3.2 An Example Hybrid Authoring Environment: CMIFed

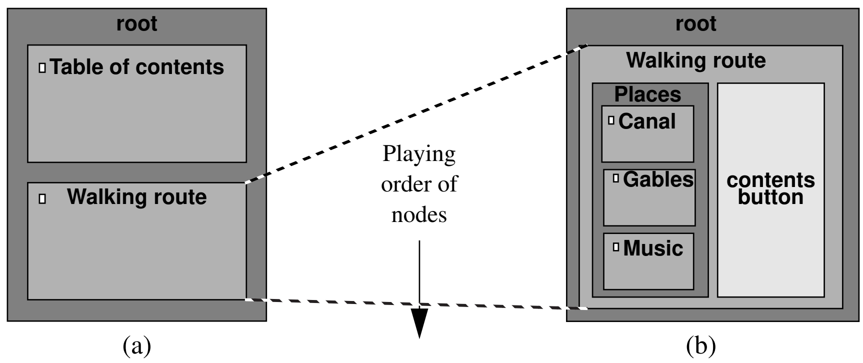
Several hybrid systems exist that combine the features of graph-, timeline-, programming- and structure-based editing. Three examples are CMIFed [30], Mbuild [17] and MET++ [1]. Of these, we look at CMIFed—which forms the core of the ESPRIT-IV authoring environment CHAMELEON [10]—in more detail to demonstrate the integration of techniques into a full-scale authoring environment.

CMIFed was designed to provide authors with a rich environment for structured viewing and manipulation of a presentation. The authoring tasks are split into three separate but closely communicating views of the presentation: the *hierarchy* view, the *channel* view and the *runtime* (or *player*) view. The hierarchy view gives the author structure-based control of the presentation. This structure can be created top down or bottom up, allowing the author to group existing media items together, or to define completely empty structure to be filled in later. The channel view is a modified timeline presentation, which describes timing information and logical resource usage. Control flow information is overlaid directly onto the channel view by using a series of synchronization constraint descriptors (called *synchronization arcs*.) The player allows the runtime view of the document to be presented, and allows user navigation using the link structure of the Amsterdam Hypermedia Model [18]. Note that no explicit programming-based view is used with CMIFed.

A presentation is composed by defining the structure of the presentation, and assigning the appropriate media items to the structure. At present, media items are of four basic types: text, still images, audio and video. Media items are created using external editor(s), available directly from within the authoring environment.

The hierarchy view for structure-based editing. The hierarchy view (Fig. 7) is the primary authoring window, providing a means of displaying and manipulating the document structure. The document has a hierarchical structure whose leaf nodes are the media items which are played in the presentation, and whose non-leaf nodes are composite nodes containing a collection of other composite nodes and/or media items. The hierarchical structure is represented in the hierarchy view as an embedded block structure. During the authoring phase, each media item in the structure diagram is assigned to a *channel*, a logical output device which is mapped by the player at runtime to a physical output device—i.e. an area on the screen or a loudspeaker.

In order to allow the author to specify timing constraints in a convenient manner, two types of composition are supported—parallel and sequential. This enables the author to group media items together to be played either at the same time or one after the other. The author does not need to specify any timing information at this point, since this is deduced from the hierarchical structure and the durations of the nodes within the structure. (Timing constraints *can* be added later—see the description of the channel view below). In Fig. 7(b) the two boxes *Places*, and *contents button* are played in parallel; the three smaller boxes nested inside the *Places* box are played one after the other. The duration of a composite node is derived by the system. The duration of a serial composite node is the sum of the durations of its children; that of a parallel composite node is the duration of the longest child. When a node has no explicit duration, for example a textual title, it is presented for the duration of its parent.



(a) The large boxes indicate different levels of structure of the presentation. The *Table of contents* part is played before *Walking route*. The small white box next to a component's name indicates that the node containing it has embedded structure not currently being shown.
(b) A zoomed-in view of the *Walking route* scene. The *Places* node contains three children each with nested structure. The right-hand box represents a text media item (a leaf node of the hierarchical structure).

Figure 7. CMIFed hierarchy view, showing top level structure of the Amsterdam tour.

The channel view for timeline-based logical resource allocation. While the hierarchy view provides a means of organizing the structure of a presentation, it provides only an indirect way of specifying logical resource use. To provide the author with an explicit time representation of the document and control of the available resources the *channel view* illustrates the media items mapped onto the available logical resources (channels). A representation of the channel view is shown in Fig. 8. By supplying this extra layer above the physical resources the author is able to describe the presentation in a system-independent way. It is up to the player software, optimized for a particular hardware configuration, to interpret the logical channels and assign the media items to the available physical output devices. A single document can be played on heterogeneous environments by adapting the players, not the document itself.

The channel view shows the timing relations derived from the structure defined in the hierarchy view. The media items making up the presentation (the leaves of the hierarchical structure) are shown with their precise durations and timing relationships. If the author changes the timing in any part of the presentation, via either the hierarchy or channel views, a new channel view is derived to reflect the change.

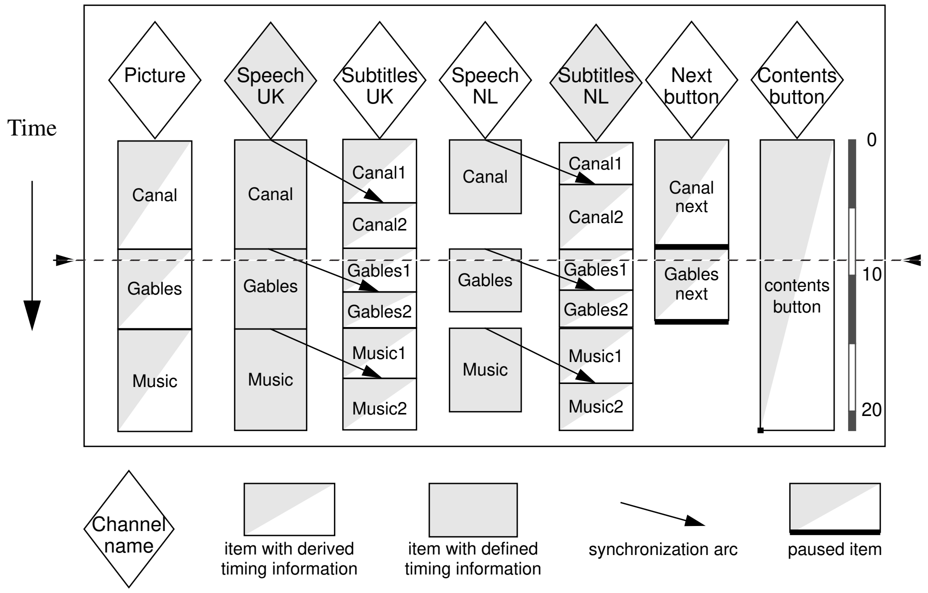
As well as providing a device-independent description of a media item's display characteristics, channels allow the author to include, for example, multiple languages (spoken or written) within one presentation rather than having to recreate the complete presentation for each language. The player allows the reader to dynamically select which language to listen to, by selectively turning channels on or off.

Creating hyperlinks. Presentations can be made interactive by providing choice points. This can be done via the use of scripts, but this leads to a navigation structure that is difficult to maintain. In order to give the author better control over the navigation structure, the *contexts* facility of the Amsterdam Hypermedia Model is used for anchor and link object grouping [19]. Creating links in a multimedia presentation is not just a matter of creating a link to a single object, but also requires support for linking to collections of items incorporating timing relations.

The CMIFed player. The player interprets the system-independent specification of the multimedia presentation (in terms of the presentation's structure, logical resource allocation and timing constraints) and plays the presentation on the available hardware. This process is described in detail in [30]. The player provides facilities, such as start, pause and stop, for the author or end-user to control the playing of the presentation.

From the authoring perspective, the player is closely integrated with the hierarchy and channel views. The player also allows users to select which channels should be played, for example to select one of a number of voice-overs in different languages.

At a low level of operation, the player converts data formats of the different media types at runtime, saving the author from having to go through tedious conversion procedures. Similarly, re-scaling the window size for the presentation is a simple operation. The window containing the channels can be scaled and the channels within will scale automatically.



The diamonds at the top of the figure show the channel names (inactive channels are shaded). The media items assigned to the channels are represented as boxes beneath the diamonds. The height of a box represents its duration. A fully-shaded box has its duration explicitly defined, either through its data type, for sound and video, or through the author assigning a specific duration. A box with a shaded triangle has inherited its duration from its parent in the presentation's structure.

Figure 8. Channel View for the *Walking route* sequence.

4 Research Challenges for Future Generations

Within the research community, several projects are underway that expand and augment present G_2 commercial approaches to providing multimedia authoring support. As with other advancements, the primary challenges that exist include improved distribution support (making documents more portable to create an attractive return on authoring investment), improved hyper-media information associations within a document (allowing full-function content-based navigation to be provided for generic media), improved media object support (allowing data to be dynamically tailored to the capabilities of the presentation environment), and support for automatic generation of document structure. We highlight a representative sample of projects that address these challenges in the following paragraphs.

4.1 Author-Once Model

Text documents (and, to a lessor degree, drawings and illustrations) benefit from using the printed page as a common model upon which to base generic authoring. Once a document is created, it can be mechanically transformed for presentation on computer screens, loose sheets of paper, books, overhead transparencies, etc., with little extra effort. Unfortunately, there is no such unifying model for multimedia. Still, the basis of multimedia authoring should be to produce a single representation of a document that can be shared among various types of presentation environments. In the multimedia case, this variety can be restricted to electronic documents, but it should span high-end to low-end computers, from fully-equipped engineering workstations to relatively simple portable computers. In supporting this heterogeneity, the guiding philosophy should be to transform a single document to fit the available presentation environment rather than creating separate versions of a document for each environment. We call this an *author-once* approach.

One approach to supporting this model is to define a ‘standard’ intermediate form, such as HyTime [22], [27] or MHEG [23], [26]. A player/program to interpret/execute standard formats on all types of platforms could then be defined. Several research projects currently exist to produce such players, although to date they have been developed in isolation from integrated authoring systems. Another approach is to define a rich document model, with explicit object resolution priorities and synchronization models that allow an author to express constraints on media object presentation. Such a model—which would encode author intentions in addition to object dependencies—could then be used to guide the instantiation of a general document on a particular hardware platform. This approach has been used in the design of CMIFed and is used implicitly in the development of document synchronization and navigation architectures ([6], [14]).

4.2 Hyper-Everything

One of the primary advantages of electronic documents (whether mono- or multimedia) is an ability to rapidly search through the information and to provide content-based, non-linear means of information navigation. In current multimedia systems, conventional hypertext facilities have been expanded to incorporate keyword-based references to audio, video and picture data. In future systems, the ability to search and 'link' pieces of information will need to become significantly more universal. While the models that support hyper-information are reasonably well understood [16], [18], methods for physically creating links and then activating them need to be improved to non-text media. The associations that are created will need to span collections of related data items and provide support for conditional activation based on the runtime state of a complex document [4], [9].

4.3 Finding and Integrating Adaptive Media Objects

Where the author-once model advocates reuse and tailoring of documents, the development of adaptive media objects supports reuse and tailoring of data items. Given the high cost of producing quality audio, video, graphic and even text data—plus the need to protect owner copyright as an incentive to increasing the number of data objects available—three aspects of object integration into documents needs to be supported: locating a particular object from an object store, extracting the relevant portion of the object for use in a particular document and transforming the representation of the fragment to meet the dynamic needs of the runtime presentation environment. The first two aspects are being addressed by research projects rooted in database systems and content-based information modelling [11], [32], [33].

Once integrated into a document, object adaptability can allow an object to be presented on a variety of systems, under a variety of hardware and performance constraints by being able to transparently transform its presentation format to the needs of the system and the users to which it is being presented. For example, a picture might be replaced by a text description depending on network resources or presentation hardware limitations; this is done to a limited degree in the Netscape Navigator World-Wide Web browser [11S]. Another type of translation may be to select one of several natural or artificial language encodings, such as substituting Dutch audio for English audio for users in the Netherlands, or for substituting computer-generation audio for text[1] [8].

4.4 Automated Document Structure Generation

Along with support for the selection and integration of media objects, further authoring support can be provided by (partially) automating the process of structural definition of the document itself. Examples of this from the intelligent interface community include [3] and [25]. A major extension to this effort is the incorporation of temporal constraints among the media objects included in the document. An approach we are investigating is to specify the subject matter to be conveyed to an end-user along with knowledge of the user's task as input to a structure generation system. Based on these constraint specifications, a document structure is generated for combining appropriate media items into coherent groups with derived temporal and layout presentation specifications.

1. An extreme example of this for the CMIFed environment was the development of a translation routine that mapped text to Morse code.

5 Conclusions

To date, the integration of multimedia capabilities on the desktop has been made possible by technological advances in processor and peripheral architectures. At some point, the emphasis will need to shift to new application and systems support tools to give the user real control over the flow of information to and through computers. We anticipate that the user requirements for more effective control over electronic information flow through local systems and across international networks will ultimately motivate new languages where fundamental support will exist for parallel, time-constrained data interaction and new operating systems will need to be defined that allow content-based, multi-stream I/O coordination, both in the context of a distributed information model that allows data to be shared and author/publisher rights to be protected.

It is difficult to predict the impact of improved electronic communication on future societies; in the short term, however, it is clear that unless grade schools integrate required drawing, video production and public speaking as adjuncts to reading and writing into the standard curriculum, creating multimedia documents will remain the domain of specialists. While there is an increase of using multimedia in teaching, teaching the skills required for effectively producing multimedia documents for general communication is in its infancy.

The capabilities of current architectures and the needs of current users should serve as a starting point in the development of authoring systems and media manipulation tools. In a sense, we are only at the stage of developing quill pens and indelible ink—and then only pens and ink that write on very limited types of paper. The emphasis for new generations of multimedia authoring systems should be geared to defining reliable means of encoding and saving information for future use. Reflecting back on the development of text-based communication, it is clear that a number of developments are still required to improve the quality, reliability, availability and cost-effectiveness of multimedia documents.

6 References

6.1 Articles

1 P. Ackermann, "Direct Manipulation of Temporal Structures in a Multimedia Application Framework," in Proceedings *Multimedia '94*, San Francisco, pp. 51 - 58, Oct 1994.

2 T. G. Aguierre Smith, "Parsing Movies in Context," in Proc. Summer *USENIX* Conference, Nashville, TN, pp. 157 - 167, 1991.

3 E. André and T. Rist, "The Design of Illustrated Documents as a Planning Task," in *Intelligent Multimedia Interfaces*, ed. Mark T Maybury, AAAI Press/MIT Press, ISBN 0-262-63150-4, pp. 94-116, 1993.

4 B. Arons, "Hyperspeech: Navigating in Speech-Only Hypermedia," in Proceedings *Hypertext '91* (Third ACM Conf. on Hypertext), San Antonio TX, pp. 133-146, Dec. 1991.

5 R. Botafogo and D. Mossé, "The MORENA Model for Hypermedia Authoring and Browsing," Proc. IEEE Int. Conf. on Multimedia Computing and Systems, Washington, D.C., pp. 40-49, May 1995,

6 M. C. Buchanan and P. T. Zellweger, "Automatic temporal layout mechanisms," in Proceedings *ACM Multimedia '93*, Anaheim CA, pp. 341-350, Aug 1993.

7 D.C.A. Bulterman, G. van Rossum and R. van Liere, "A Structure for Transportable, Dynamic Multimedia Documents," in Proc. Summer *USENIX* Conference, Nashville, Tennessee, pp. 137-155, 1991.

8 D.C.A. Bulterman, "Specification and Support of Adaptable Network Multimedia," ACM/ Springer *Multimedia Systems*, 1(2), pp. 68-76, September 1993.

9 V.A. Burrill, T. Kirste & J.M. Weiss, "Time-Varying Sensitive Regions in Dynamic Multimedia Objects: A Pragmatic Approach to Content-Based Retrieval from Video," *Information and Software Technology,* 36(4), Butterworth-Heinemann, pp. 213-224, April 1994.

10 CHAMELEON: An Authoring Environment for Adaptive Multimedia Documents, CEC ESPRIT-IV Project 20597, 1995.

11 M. Davis, "Media Streams: An Iconic Language for Video Annotation," *Cyberspace*, 84(9), (Norwegian Telecom Research), pp. 49-71, 1993.

12 G.D. Drapeau and H. Greenfield, "MAEstro — A Distributed Multimedia Authoring Environment," in Proc. Summer 1991 *USENIX* Conference, Nashville, TN, pp. 315-328. 1991.

13 S. Eun, E.S. No, H.C. Kim, H. Yoon and S.R. Maeng, "Eventor: An Authoring System for Interactive Multimedia Applications," *Multimedia Systems* 2(2), pp.129 - 140, 1994.

14 K. Fujikawa, S. Shimojo, T. Matsuura, S. Nishio and H. Miyahara, "Multimedia Presentation System 'Harmony' with Temporal and Active Media," in Proc. Summer 1991 *USENIX* Conference, Nashville, TN, pp. 75-93, 1991.

15 D. J. Gemmell, H.A. Vin, D.D. Kandlur, P.V. Rangan and L.Rowe, "Multimedia Storage Servers: A Tutorial," IEEE Computer, 28(5), pp. 40-49, May 1995.

16 Frank Halasz and Mayer Schwartz, "The Dexter Hypertext Reference Model," *Communications of the ACM*, 37 (2), pp. 30 - 39, Feb. 1994 (Also NIST Hypertext Standardization Workshop, Gaithersburg, MD, January 16-18 1990.)

17 R. Hamakawa and J. Rekimoto, "Object Composition and Playback Models for Handling Multimedia Data," *Multimedia Systems* 2(1), pp. 26-35, 1994.

18 L. Hardman, D.C.A. Bulterman and G. van Rossum, "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model," *Communications of the ACM*, 37 (2), pp. 50 - 62, Feb. 1994.

19 L. Hardman, D.C.A. Bulterman and G. van Rossum, "Links in Hypermedia: The Requirement for Context," in Proceedings *ACM Hypertext '93*, Seattle WA, pp. 183-191, 1993.

20 L. Hardman, G. van Rossum and D.C.A. Bulterman, "Structured Multimedia Authoring," in Proceedings *ACM Multimedia '93*, Anaheim CA, pp. 283 - 289, Aug. 1993.

21 M.E. Hodges, R.M. Sasnett and M.S. Ackerman, "A Construction Set for Multimedia Applications," *IEEE Software,* 6(1), pp. 37-43, Jan. 1989.

22 ISO, "HyTime. Hypermedia/Time-Based Structuring Language," ISO/IEC 10744:1992.

23 ISO, "MHEG. Information Technology Coded Representation of Multimedia and Hypermedia Information Objects (MHEG) Part 1: Base Notation (ASN.1), ISO/IEC CD 13552-1:1993, Oct 15 1994.

24 J.F. Koegel and J.M. Heines, "Improving Visual Programming Languages for Multimedia Authoring," *ED-MEDIA '93*, World Conference on Educational Multimedia and Hypermedia, Charlottsville, Virginia, pp. 286-293, June 1993.

25 R. Macneil, "Generating Multimedia Presentations Automatically Using TYRO, The Constraint, Case-Based Designer's Apprentice," in Proceedings: *IEEE 1991 Visual Language Workshop*, pp. 74-79, 1991

26 T. Meyer-Boudnik and W. Effelsberg, "MHEG Explained," *IEEE MultiMedia*, 1(4), pp. 26-38, Spring 1995.

27 S.R. Newcomb, N.A. Kipp and V.T. Newcomb, "HyTime: The Hypermedia/Time-Based Document Structuring Language," *Communications of the ACM*, 34(11), pp. 67-83, 1991.

28 R. Ogawa, H. Harada and A. Kaneko, “Scenario-Based Hypermedia: A Model and a System,” in Proceedings: *ECHT ’90* (First European Conference on Hypertext), INRIA France, pp. 38-51, Nov. 1990.
29 S. Travis Pope and G. van Rossum, “Machine Tongues XVIII: A Child’s Garden of Sound File Formats.” *Computer Music*, 19(2), pp. 25-63, 1995.
30 G. van Rossum, J. Jansen, K.S. Mullender and D.C.A. Bulterman, “CMIFed: A Presentation Environment for Portable Hypermedia Documents,” in Proceedings *ACM Multimedia ’93*, Anaheim CA, pp. 183-188, Aug. 1993.
31 A. Siochi, E.A. Fox, D. Hix, E.E. Schwartz, A. Narasimhan, W. Wake, “The Integrator: A Prototype for Flexible Development of Interactive Digital Multimedia Applications,” *Interactive Multimedia*, 2(3), pp. 5-26, 1993.
32 H. Wittig and C. Griwodz, “Intelligent Media Agents in Interactive Television Systems,” Proc. IEEE Int. Conf. on Multimedia Computing and Systems, Washington, D.C., pp. 182-189, May 1995.
33 H. Zhang, Y. Gong, S. Smoliar, “Automatic Parsing of News Video,” Proc. First IEEE ICMS, Boston, 1994.

6.2 Software

1S Adobe Systems, Inc., “Adobe Photoshop 3.0.” `URL:http://www.adobe.com/`
2S Adobe Systems, Inc., “Adobe Premiere.” `URL:http://www.adobe.com/`
3S Apple Computer Corp., “HyperCard.” `URL:http://www.apple.com/`
4S J. Bradley, “XV,” University of Pennsylvania. Available via anonymous `ftp: ftp.cis.upenn.edu:pub/xv`
5S IBM Corp., “AVC: Audio Visual Connection,” User’s Guide, 1990.
6S Frame Technology Corp, “FrameMaker 5.0.” `URL:http://www.frame.com/`
7S Kaleida Labs, Inc., “ScriptX.” `URL:http://www.kaleida.com/`
8S Macromedia, Inc., “Macromedia Director 4.0 for Windows.” 1994. `URL:http://www.macromedia.com/`
9S Macromedia, Inc., “Macromedia SoundEdit 16.” 1993 `URL:http://www.macromedia.com/`
10S Microsoft Corp., “Microsoft Word 6.0.” 1994. `URL:http://www.microsoft.com/`
11S Netscape Comm. Corp, “Navigator 1.1,” 1995. `URL:http://home.mcom.com/`