Authoring Support for Durable Interactive Multimedia Presentations

Lynda Hardman and Dick C.A. Bulterman

CWI
Multimedia Kernel Systems Project
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Email: {Lynda.Hardman, Dick.Bulterman}@cwi.nl

Abstract

There are two major problems with the current ways of creating interactive multimedia presentations. Firstly, authoring a multimedia presentation is a non-trivial task, requiring a range of skills such as creating individual items in each medium as well as combining these into a coherent presentation. Secondly, after having devoted a large amount of time and effort to the creation of a presentation, there is no guarantee that it can be played back on a platform other than the one for which it was created, let alone whether it can be played back by future systems. To tackle both these problems, we first present an information model for interactive multimedia, so that information can be stored independently of the system that creates or plays it. We then investigate a number of authoring systems. By differentiating four authoring paradigms we classify and describe a selection of both research and commercial systems. These provide examples of the types of support that can be given to authors, and how this support can be provided in practice. Using the approaches and features supported by these systems as a base, we give an analysis of the facilities desired in an ideal authoring system.

1. Introduction

A large number of multimedia authoring systems exist as either academic prototypes or as readily available commercial systems. They each allow the creation of a multimedia presentation from a selection of media items using the system's own proprietary document format and its own suite of tools for creating documents conforming to this format. The use of proprietary document formats leads to the creation of presentations that can be played back only by the system with which they were created. A broad range of tools for carrying out similar tasks leads to non-ideal ways of creating a presentation.

Assembling media items into a presentation is part of an overall process of creating a multimedia presentation. We are interested in designing a system that eases the assembly process for the author. We are less concerned, here, with the creation of individual data items (for example using specialist data editors for text, images, video, sound, animation etc.), nor are we focussing on the data formats used, nor the necessary trade-offs for authoring presentations destined to be played over a network. The focus of this STAR report is to propose a data model that captures sufficient information for recreating an interactive multimedia presentation, and to propose the requirements for a suite of easy to use tools that can create presentations conforming to the model.

The essence of multimedia authoring is that the author wants to communicate a message to the reader. In order to achieve this, the author is required to specify the timing and placement of objects in the presentation, as well as giving navigation choice points. In some respects, authoring multimedia can be compared with formatting text. Both activities require the collection/generation of source material and the placement of these sources within a presentation environment. A generic text formatter allows an author to layout information for use on a printed page. Depending on the features supported by the formatter, authors may be able to vary the font and size of the text, they may be able to vary the spatial layout of the information on the page, and they may be able to incorporate higher-level structures (such as chapters and sections or figures and tables) in the document. In the same way, multimedia authoring tools allow a user to integrate several types of information into a composite presentation. Unlike text, however, the temporal dimension often dominates the authoring process. In many respects, then, multimedia authoring is more akin to the definition of dynamic events such as dance choreography [7]. A choreographer wishes to design a dance that can be performed by a number of dancers; each dancer performs her/his own movements at a particular place on the stage, in some time relation to the music; the dancers sometimes dance together and sometimes independently. The parallel with a multimedia presentation is that the stage is the computer screen; the music is a time base along which movements can be synchronized; and a set of movements is a media object, which can be "played" by a dancer.

This report is structured as follows. We introduce an example interactive multimedia presentation and propose our data model for describing the elements required for describing a platform-independent interactive multimedia presentation. This is followed by descriptions of a comprehensive selection of existing and influential commercial multimedia authoring systems. Section 4 gives an analysis of the requirements for the ideal multimedia authoring system, referring back to the previous system descriptions for illustrative examples. The report ends with some conclusions and thoughts on possible, desirable and probable futures for multimedia authoring systems.

2. A Document Model for Interactive Multimedia Presentations

As a starting point for our discussion on authoring systems, we consider the characteristics of a "typical" multimedia presentation. One example of such a presentation is shown in Fig. 1, which illustrates three fragments from a tour of the city of Amsterdam. In the top fragment, which is analogous to a table of contents, a user is given a description of the tour and a number of alternatives that can be selected within the presentation. One of these alternatives is illustrated—containing a description of walks through the city, highlighting several features found on the tour, which is itself sub-divided across a number of other fragments. From a media perspective, each fragment consists of a number of media items displayed on screen or played through loudspeakers.

A multimedia presentation can be displayed in the manner of a film—a fixed sequence of information fragments that are projected to a passive user. Alternatively, it can contain choice points, as illustrated in the figure, with which end-users can interact. Multimedia documents which support this style of interaction via navigation are known as *hypermedia* documents. In a hypermedia document, the end-user can choose to follow a navigation path through the information via *links*.

In order to enable the description of a multimedia document for presentation on multiple heterogeneous platforms a document model is required which can be interpreted by a presentation system for that platform. This requires the inclusion of sufficient information to reproduce the presentation on different environments and the encoding of this representation in a platform-independent form. We present our model for multimedia documents including interaction via hypertext links.

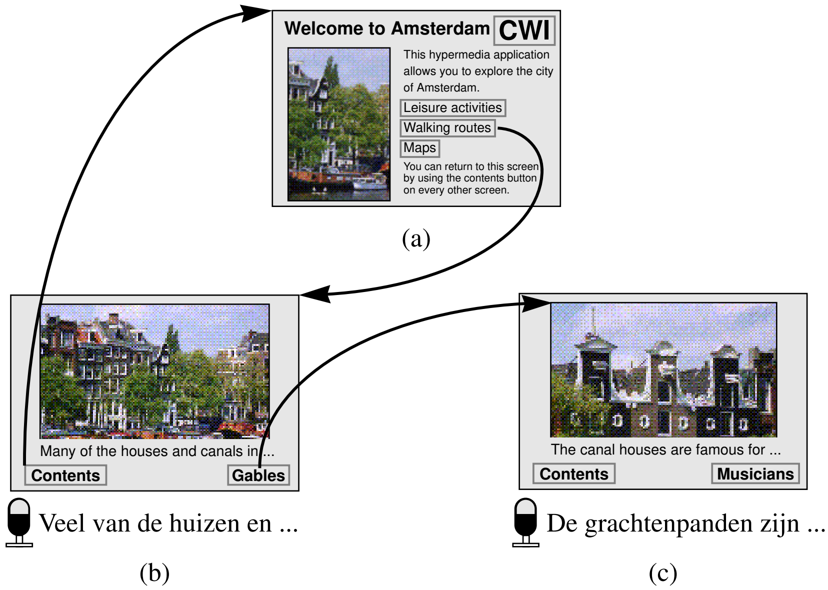


Figure 1. An example interactive multimedia presentation.

2.1. The Amsterdam Hypermedia Model

An important principle guiding the development of usable authoring systems is a sound data model around which editing tools can be built [17]. Models for multimedia and hypertext already exist, where the former describe timing relations and the latter structural relations. Neither is able to express the full complexity of a hypermedia presentation—a combination of multimedia timing and layout, and hypertext structuring. By using a model which captures the author's intentions, rather than platform and data-dependent presentation details, and interpreting these at playback the author is spared work and the presentation can be played on a range of hardware platforms.

In order to provide such a model for hypermedia, four fundamental types of relationships require to be described in addition to the media items included in the presentation: structural, timing, layout and interaction relations. Structural relationships define logical connections among items, including the grouping of items to be displayed together and the specification of links among these groupings. Timing relations specify the presentation dependencies among media items, possibly stored at different sites. Layout specifications state where screen-based media are to be sized and placed, either in relation to each other or to the presentation as a whole. Interactions, through navigation, give the end-user the choice of jumping to related information.

In order to capture sufficient information for specifying a hypermedia presentation we need to look at models of both multimedia and hypertext, and combine the information required by both paradigms. The Dexter hypertext reference model [11], developed by a group of hypertext system designers, describes the required structural relations. Our previous work [5] describes a model for specifying timing relations between collections of static and dynamic media composed to form multimedia presentations. The Amsterdam Hypermedia Model (AHM) [14] builds on these two bases and presents a model for structured, multimedia information. A diagrammatic impression of the AHM is given in Fig. 2 which we will refer to in the following subsections describing the main elements of the model, with emphasis on their relevance for a hypermedia author.

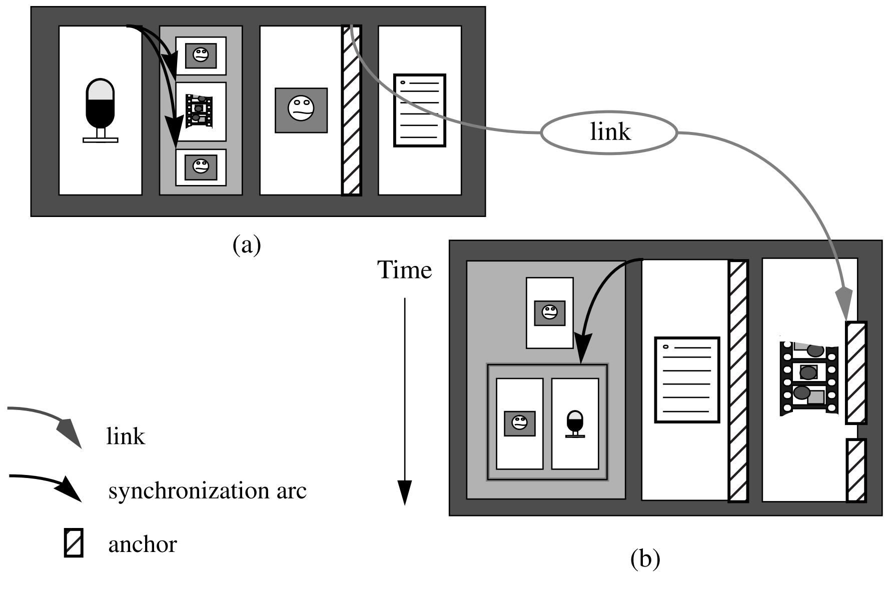
2.1.1. Timing information

Timing relations are normally the first that come to mind when building a multimedia presentation—to the extent that the timeline (see Fig. 3) is often used as the basis of an authoring paradigm (discussed fully in section 3). Timing information is needed to express when each of the media elements composing the presentation will appear on (and disappear from) the screen. This information can be given by specifying a time when an element should appear relative to the complete presentation—e.g. 30 seconds after the beginning of the presentation a subtitle corresponding to the spoken commentary should appear—or relative to other elements being played in the presentation—e.g. 3.5 seconds after the spoken commentary begins the subtitle should appear. The advantages and disadvantages of either of these approaches is not important for the model, the point is that both possibilities need to be expressible in a model of multimedia information. The AHM allows the specification of timing relations defined between single items, groups of media items or between a single item and a group. For example in Fig. 2(a) both delays are specified between two media items, in this case so that the video and image correspond to the spoken commentary. In Fig. 2(b) the delay is specified from a text item to the beginning of a grouped image and spoken commentary. (Note that the model specifies no boundaries on these timing relations—it is possible to define a negative delay, so that an item starts before the whole group starts.) These timing relations are specified in the model as *synchronization arcs*. These can be used to give exact timing relations, but can also be used to specify more flexible constraints, such as “play the second media item 3 +/- 0.3 seconds after the first”.

2.1.2. Structural information

The structure items of the model, including composition, links and anchors illustrated in Fig. 2, are described here briefly.

- *Composition* plays an important role in multimedia presentations, where almost all presentations contain more than one media element. A composition structure allows a group of media items to be created which can then be treated as a single object. For example, scenes in a presentation can be re-ordered by manipulating objects representing the scenes (for example the outer boxes in Fig. 2(a) and (b)), rather than having to re-order the individual items composing the scenes. Composition is also useful for building up scenes from smaller groupings of information. For example, a company logo can be grouped with a spoken commentary (e.g. the image



The main components of the model are: *media item* (white box), *composition* (shaded boxes), *link, anchor,* and *synchronization arc.* See text for explanations.

Figure 2. Amsterdam Hypermedia Model.

and commentary in the lower left part of Fig. 2(b)) and included in several places in the overall presentation. The composite structure is used to store the timing information specified among its constituent elements (for example the synchronization arcs also shown in the figure).

- *Anchors* are a means of indexing into the data of a media item, allowing a part of the item to be referred to in a media-independent manner. The data specified by an anchor can be used as the beginning or end marker for a link (see following paragraph). For example, within text, an anchor might define a character string; in an image an area on the screen. In continuous media, such as video or animation, the area on the screen may change with time [6]. Indeed, the link marker may occur only for part of the duration of the media item—for example the video item in Fig. 2(b) shows two anchors which are displayed at different times. One of the uses of a dynamic link marker is to follow moving objects within the video or animation.
  In a composite item, for example the scene represented by Fig. 2(b), an anchor can refer to the scene as a whole (allowing navigation to the complete scene) or indirectly to anchors in the media items. For example, a video of a bouncing ball may be accompanied by a text about the ball. Both the moving image of the ball and the word *ball* in the text are part of the same (composite) anchor. The anchor in the composite representing the concept "ball" refers to two anchors—the anchor in the text item (the striped box next to the text item in Fig. 2(b)) and one of the anchors in the video item (the upper of the two video anchors in the figure). When the author wishes to relate other information to the ball's description (e.g. how gravity affects bouncing) then a link can be made to the composite "ball" anchor.
- *Links* enable the end-user to jump to information deemed by the author to be relevant to the current presentation. The reader can follow a link by selecting an active link marker (specified via an anchor) on the screen—often by pointing with a mouse and clicking. The link can lead to another presentation or a different part of the same presentation, as illustrated in Fig. 1. The links specify, declaratively, interaction choices the reader can make during the presentation. This form of interaction is very common in multimedia systems, but is very often specified using "goto" commands attached to a highlighted area on the screen (which is not necessarily bound to the data item it refers to).
  The link structures, similar to the synchronization arcs, can be created between media items, or groups of items. (Note, however, that the structures are used for very different purposes: links make semantic connections between items; synchronization arcs specify timing relations, which are similar in status to layout relations.)
- *Context for links* is needed for specifying which media items on the screen are affected when a link is followed. For example, in a presentation composed of multiple media items, the complete presentation may be replaced (illustrated in Fig. 2(a) to (b)) or only a part (e.g. Fig. 2(b) to (c)). There is also a choice of whether the information at the destination of the link replaces the scene that was being played, or whether it is displayed in addition. In the figure the destination replaces the source in both cases. When the current presentation also contains one or more continuous media items there is a further choice as to whether the current scene should stop or continue playing when the link is followed.
  Adding navigation structures to multimedia presentations through the use of link contexts is discussed in greater detail in [15].

2.1.3. Layout information

Layout information can be attached to each media item in the model, but this is normally done through the use of *channels,* which specify re-usable presentation information. For example, the images in Fig. 1(b) and (c) were created with different sizes, but they are both scaled to make use of the same screen area, specified by the channel they are assigned to. Similarly, the subtitles make use of the same text channel, which specifies the font style and size to be used, as well as the positioning on the screen. The use of channels is further discussed in the description of the authoring environment CMIFed (section 3.1.1).

2.1.4. Summary

The Amsterdam Hypermedia Model has been designed to capture the elements of structure, timing, layout and interaction that are needed to specify an interactive, time-based multimedia presentation. The model does not prescribe the form these relationships should take, but implies that they should exist, and proposes how to combine them with each other. Once a presentation has been described using our model, its description can be expressed in different intermediate forms suitable for multiple presentation environments. While we do not prescribe a language for specifying a presentation conforming to our model, it could be expressed in a system-independent language such as the HyTime (Hypermedia/Time-based Document Structuring Language) international standard [20], [28][1], based on SGML (Standard Generalized Mark-up Language) [33]. Similarly, transmission of (groups of) multimedia objects conforming to the model could be carried out using a standard such as MHEG (Multimedia and Hypermedia Expert Group), [25], [26].

3. Approaches To Authoring Interactive Multimedia Documents

We describe a number of authoring tools in order to demonstrate the variety, and similarities, of different approaches. Most of the tools discussed are to be found in the academic literature, since they explore the more innovative approaches. We have also chosen, however, to include a number of commercial systems, since these have survived a number of years of use by real authors. While a large selection of commercial authoring systems is available ([18], [21], [34], [36][2]) we have chosen Authorware, IconAuthor and Director as being representative of the more widely-used, and thus better, authoring applications.

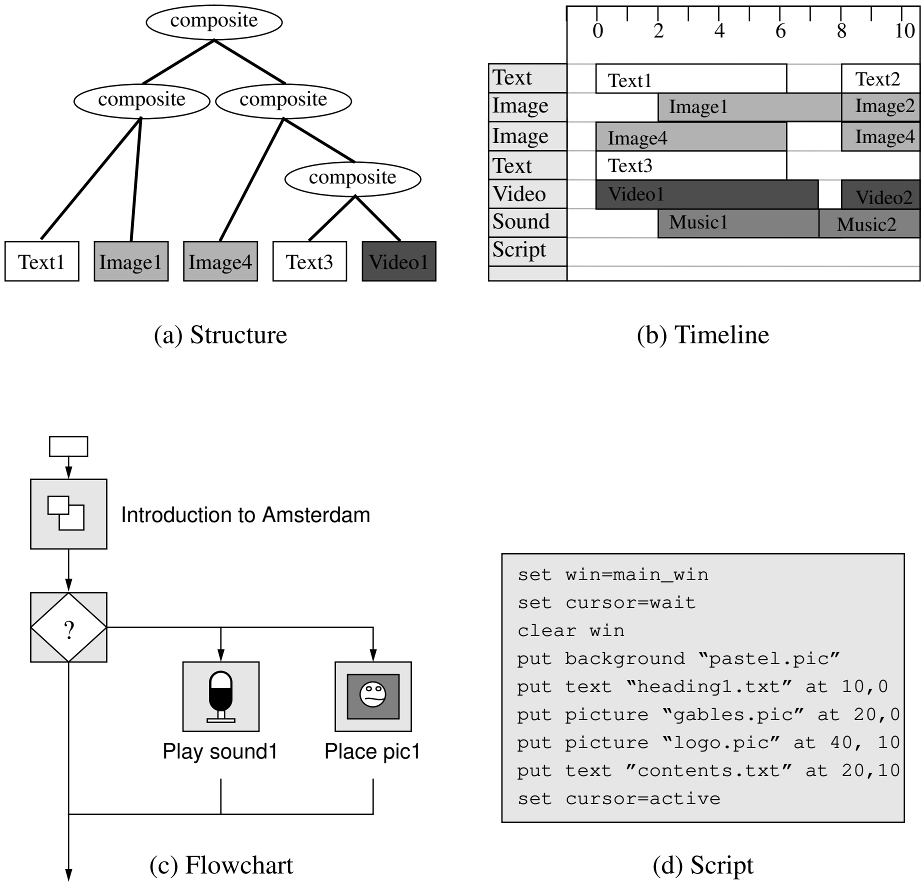
A number of different underlying paradigms are supported by multimedia authoring systems. The majority of systems fall under the following: *structuring*, *timeline*, *flowchart* and *script*. A diagrammatic impression of these paradigms is given in Fig. 3.

- *Structure-based* authoring systems support the explicit representation and manipulation of the structure of a presentation. This gives the advantage of being able to group items together into "mini-presentations" which can be manipulated as a whole. Another advantage can be given by deriving the timing relations in the presentation from the structure, so that alterations in durations of objects are propagated through the presentation by the system.
- *Timelines* show the constituent media items placed along a time axis, possibly on different tracks. These are useful for giving an overview of which objects are displayed on the screen, when.
- A *flowchart* gives the author a visual representation of the commands describing a presentation. While systems using this approach are deemed simpler to use, they tend to become unwieldy for large presentations.

---

1. Note that the details in the published standard, [20], have superceded those described in [28].

2. Systems described in these articles: *Act III*, Informatics Group; *Aldus SuperCard 1.6*, Aldus Corporation; *Animation Works Interactive 1.1*, Gold Disk; *Ask*Me Pro 2.12*, Ask*Me Information Center; *Authorware Professional 2.0.1*, Macromedia; *AVC* 1.05, IBM; *Course Builder 4.0*, Discovery Systems International; *HSC InterActive 1.0*, HSC Software; *HyperCard 2.1*, Claris Corporation; *IconAuthor 5.1*, AimTech; *LinkWay Live!*, IBM; *Director 3.1*, Macromedia; *Multimedia Desktop*, Datalus; *Multimedia Toolbook 1.5*, Asymetrix; *Quest 4.0*, Allen Communication; *Spinnaker Plus 2.1*; Spinnaker Software; *Storyboard Live!*, IBM; *Test Factory 2.1*, Warren-Forthought.



(a) A *structure-based* representation allows manipulation at the scene level.
(b) A *timeline* representation gives the author a clear overview of which objects are playing when during the presentation. Manipulation takes place at the object level.
(c) A *flowchart* shows the structure of the presentation clearly. The order of play can be deduced, but time is not represented explicitly.
(d) A *scripting language* gives the author the flexibility of specifying every action on the screen. There is no explicit representation of time or scene structure.

Figure 3. Example authoring paradigms.

- A *script-based* system provides the author with a language where positions and timings of individual objects can be specified. Although scripting languages provide a flexible authoring interface, they have the disadvantage of becoming unmanageable in large presentations. Structures such as scene boundaries or timing relations between media items are difficult to recognise in the script.

3.1. Structure-based Authoring Systems

3.1.1. CMIFed

We discuss our own authoring environment, CMIFed, in detail to give a feel for a full structure-based authoring environment. CMIFed has been 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, channel view and player. The *hierarchy view* gives the author control of the structure of the presentation. There are no strict divisions between "scene" or "sequence"; rather, an overall hierarchical structure is provided that can be viewed and edited at different levels of detail. 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 (this approach is also taken in Mbuild [12], discussed in section 3.1.4).

The document format used by the CMIFed system corresponds closely to the theoretical Amsterdam Hypermedia Model. The structural information defined by the author in the hierarchy view is used to derive basic timing information for the presentation. This timing information, along with other logical resource usage, is displayed in the *channel view*, where extra synchronization constraints between any two media items can be created. The *player* allows the author to see the presentation as the readers will see it. These three views of the presentation are described below. More detail on the interaction methods in each view can be found in [16].

The hierarchy view for structure manipulation

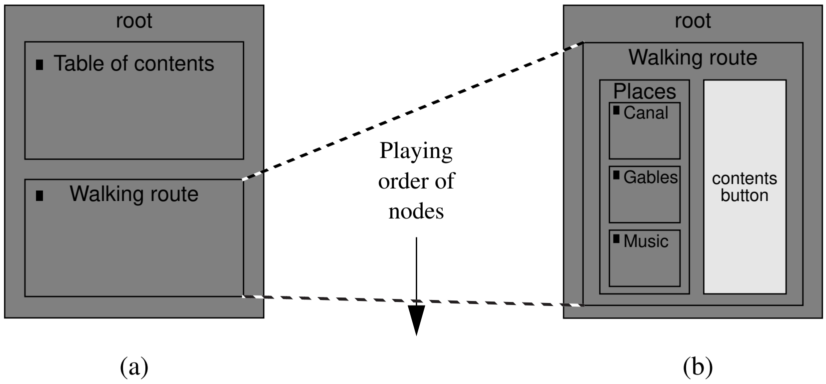
A presentation is composed by defining the structure of the presentation, and assigning the appropriate media items to the structure. At present, media items are of four basic types: text, still images, audio and video. Media items are created using external editor(s), available directly from within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

The hierarchy view (Fig. 4) 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. Each media item 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.

The structure of the Amsterdam tour is shown in Fig. 4. The *Table of contents* corresponds to the screen in the upper part of Fig. 1 and *Walking route* section contains the two clips shown in the lower part of Fig. 1. The author can navigate around the hierarchical structure by zooming in and out of the nested structures. This not only reduces the screen space required for representing the structure, but gives a focussed view of the part of the structure the author is currently interested in. For example, if we zoom in on the *Walking route* in Fig. 4(a), we see the structure shown in Fig. 4(b).

Authoring work can be reduced through the use of the hierarchical structure, for example screen layouts can be designed where persistent objects, such as titles and logos, need be placed only once and are retained on the screen throughout the scene. For example, the text node in Fig. 4(b) remains on the screen for the duration of the *Walking route* sequence.

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. 4(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, reducing the amount of information the author is required to specify. 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 similar technique, called glue in Mbuild, is described in [12] and section 3.1.4).



(a) The large boxes indicate different levels of structure of the presentation. The *Table of contents* part is played before *Walking route*. A small dark box 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 shaded right-hand box represents a text media item (a leaf node of the hierarchical structure).

Figure 4. CMIFed hierarchy view, showing top level structure of the Amsterdam tour.

A multimedia presentation can be authored by first creating the structure and then assigning media items at the leaves of the structure. Alternatively already existing (media) nodes can be grouped together into higher levels of structure. A media item needs to be assigned to a channel (a logical output device) and to have a data object associated with it, usually by a reference to a file containing the data. When associating a file with a media item, the author does not need to be aware of the details of the data format of the element being inserted, since the player will convert it (if it is one of the recognized formats) at runtime.

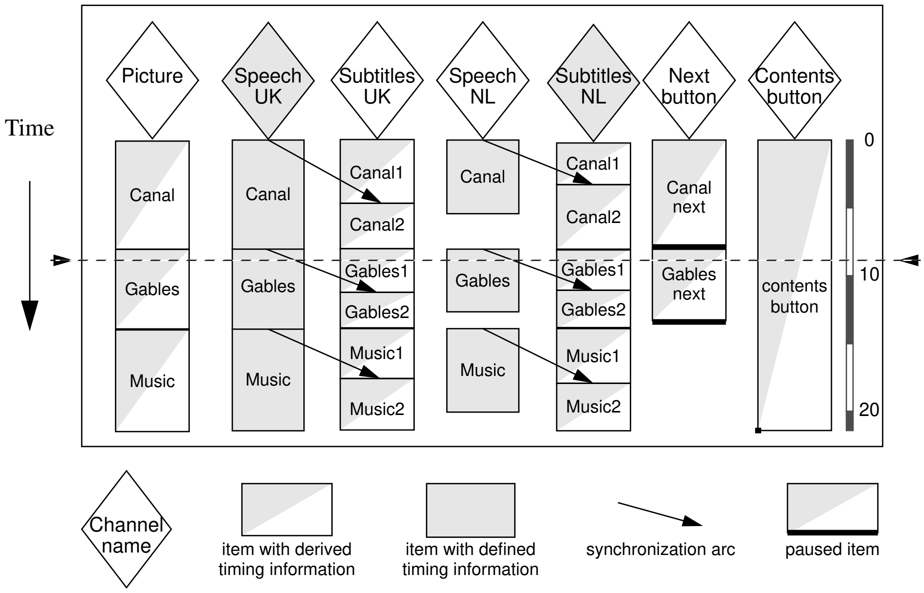
The channel view for logical resource allocation

While the hierarchy view provides a means of organising 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* provides a view of the media items mapped onto the available logical resources (channels). 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.

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, the channel view is immediately updated to reflect this. The correspondence between the structure shown in the hierarchy view (Fig. 4) and the media items in the channel view (Fig. 5) is shown in Fig. 6.

The channels enable the author to define high-level presentation characteristics for each media type, so that presentations can be composed without having to specify details for each individual node: for example, a sound channel defines a volume; a text channel defines a rectangular area on the screen and a font. Attribute values can be overridden by an individual media item, for example, a short text string can be displayed in a larger font. The media items are each assigned to a channel.

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



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 5. Channel View for the *Walking route* sequence.

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.

The ordering of media items during the multimedia presentation is taken directly from the channel view—time runs from top to bottom and nodes on all the active channels are played in parallel. For example, the media items intersecting the dotted, horizontal line in Fig. 5 are those presented during the gables clip shown in Fig. 1.

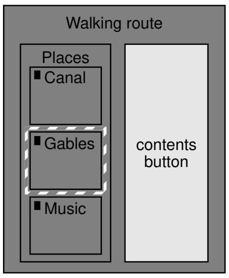
While the majority of the timing relations among media items are derived satisfactorily from the hierarchy view, the author may wish to explicitly state some timing constraints. The author is able to do this by creating synchronization arcs between media items. For example, the display of the text line of the *Gables2* subtitles in Fig. 5 coincides with the utterance of the associated phrase, somewhere within the *Gables* audio object.

In addition to the authoring facilities available from the channel view, the state of the system is shown by dynamically highlighting the media items as they are being played. When the system has sufficient time it looks ahead in the presentation and fetches data that will be needed. The stages of this pre-scheduling are also shown by highlighting the media items in the channel view. Further details on the scheduling are given in [21].

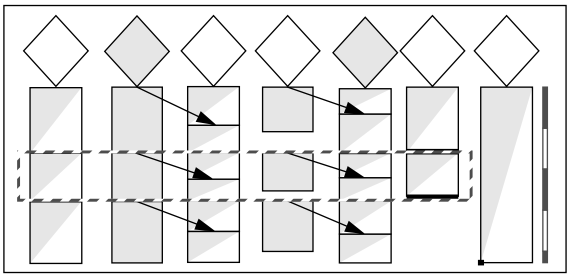
Creating hyperlinks

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, we use the Amsterdam Hypermedia Model of anchor and link objects [14].

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.



(a) Figure 3(b)



(b) Figure 5

Media items contained in the nested structure of the *Gables* section in (a) correspond to the media items enclosed by the stippled box in (b).

Figure 6. Correspondence between hierarchy and channel views.

For example, in Fig. 1 the *Walking routes* label, in the upper part of the figure, takes the reader to a complete scene, with descriptions of the timing relations among the objects being played—following the link from this label replaces the contents screen with a complete new scene. It is not always the case that the whole scene needs to be replaced, for example, following the link from the *Gables* label, in the lower half of the figure, replaces only part of the structure (shown as the *Gables* box in Fig. 4), while the *Contents* text remains unaffected on the screen (the *contents button* box in Fig. 4).

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 [31]. The player provides facilities, such as start, pause and stop, for the author or end-user to control the playing of the presentation. Fig. 1 shows three scenes in a presentation as they appear in the player.

From the authoring perspective, the player is closely integrated with the hierarchy and channel views. This allows the author to play any part of the presentation, from one node through the different levels of structure to the whole presentation, providing an (interactive) indication of how the presentation will appear to the end user. Note that by adjusting the scope of the presentation fragment, the author can preview a small section of the presentation without having to play the entire sequence.

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. This is illustrated in Fig. 1, where the reader has selected to listen to Dutch speech and read English subtitles. The corresponding channels can be seen in Fig. 5.

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, rescaling the window size for the presentation is a simple operation. The window containing the channels can be scaled (e.g. the large rectangles in Fig. 1) and the channels within will scale automatically.

3.1.2. Athena Muse

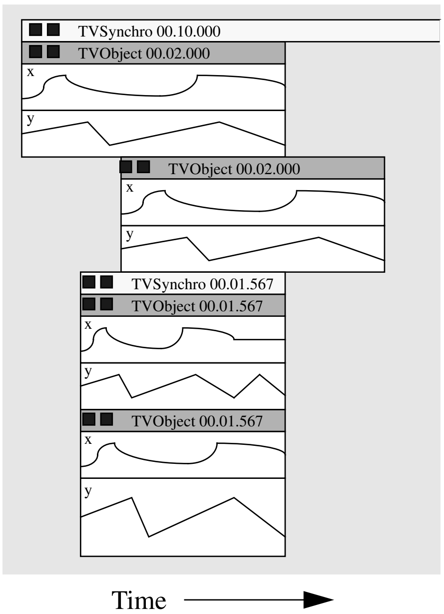
The Athena Muse system [17] combines four representation schemes for the construction of multimedia educational software. These do not correspond exactly to the four paradigms we discuss, but are described as directed graphs (a concept-based structure), multi-dimensional spatial frameworks (generalisations of timelines), declarative constraints and a procedural language (script). The first two are used to describe hypermedia documents through the use of a directed graph of packages, where a package can be considered as a small multimedia presentation consisting of text, video and graphics objects (in our terminology a multimedia document). The packages are nodes in a directed graph, with arcs representing possible transitions between the packages. The arcs can be used to represent hypermedia links, where cross references are fired by sending activation signals from one package to another. The network is built on a concept structure with links between high-level abstractions (how these are translated to the package level is not explained).

Each package can be described in an N-dimensional space, where, for the case of multimedia, three dimensions are sufficient for describing time and layout. An example is the use of a timeline for attaching subtitles to a video sequence. This is done by introducing a timeline and synchronizing both the video sequence and the subtitles with respect to the timeline. The authors argue that this makes it easy to add and remove channels of information without disrupting the whole presentation. We, however, feel that the synchronization conditions should be expressed directly between (parts of) the media items themselves, so that if, for example, the video sequence is shortened or lengthened the subtitles stay synchronized with the correct part of the video.

The structuring of a package does not continue down to the sub-package level (this is expressed in the N-dimensional space), which prevents timing relations from being derived from the structure. Neither does it extend above the package level to group sets of packages together, which prevents the structure from being used as a high-level storyboard by the author.

3.1.3. MET++

In the MET++ application framework [1] a multimedia presentation is considered to be a hierarchy of serial and parallel compositions of media items. The temporal layout of the constituent items is derived from this composition hierarchy automatically. The building blocks consist of time layout objects and media objects, where each has a starting point, a duration and an associated virtual timeline. These are incorporated into a hierarchical structure with the media objects as leaf nodes and the time layout objects as intermediate nodes. When the start time or duration of an event is



Each *TVObject* represents a media item and specifies its *x* and *y* positions. *TSynchro* titles are parallel composites of the objects below them. Both types of objects can be cut, copied, and pasted, stretched and shrunk. (We are unable to give an example of serial structuring, since this is not shown in the article.) (Copied without permission from Fig. 7, [1].)

Figure 7. Time composition view in *MET++*.

altered all time positions are recalculated (as happens in CMIFed). The timeline representation (Fig. 7) allows the visualization and manipulation of the hierarchical structure (whereas these are separated in CMIFed). Any object, media item or composition, can be stretched or reduced in time. In the case of a composite object, the transformation is applied throughout the hierarchy.

The timeline representation of the multimedia presentation (Fig. 7) shows clearly the values of attributes that vary over time—the horizontal and vertical positions in the figure. This representation could be used for other object parameters, e.g. the volume of an audio object, or the fade-in rate of an image or video.

There is no concept of stream, track or channel in MET^{++} — each object is played at its own specific (not necessarily static) location for the duration specified on the timeline. This enables the structure to be shown in the timeline view, since there is no restriction that the grouped objects require to be placed on adjacent channels or tracks, as is the case in CMIFed and the Integrator which make use of channel and track structures, respectively.

3.1.4. Mbuild

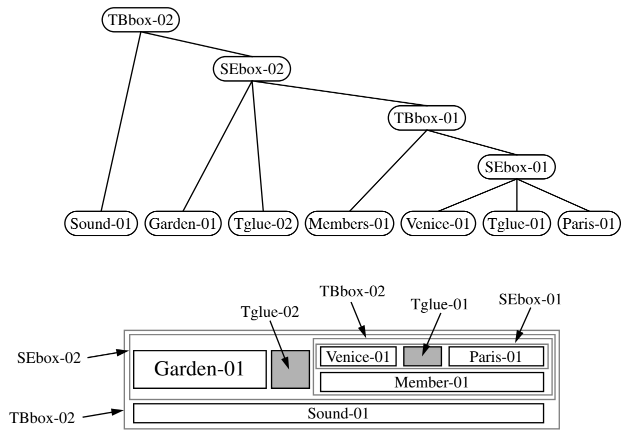
The multimedia editing system *Mbuild* [12] uses a hierarchical structure of composites and multimedia items for editing and reusing composite multimedia data. The timing of the presentation is determined when the highest ranking composite object is determined. The timing is calculated using a feature called temporal glue (which is similar to the method used in the CMIFed). Multimedia authors are able to create empty hierarchical structures reflecting the narrative of the presentation and only later fill them with the desired media items, which is also similar to a use of the hierarchy view in CMIFed.

3.2. Timeline-based Authoring Systems

Timeline based authoring systems allow the placement of media objects on a timeline. If the start time or duration of a media item is changed (e.g. by editing the data belonging to the item) then the positions of the other elements on the timeline remain the same, requiring the author to edit them explicitly.

3.2.1. Director

Director [36] is a commercial system 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



The upper part of the figure shows the hierarchical structure of a presentation. The lower part shows the equivalent structure as displayed in *Mbuild*. (Copied without permission from Fig. 20, [12].)

Figure 8. Hierarchical structure in *Mbuild.*

timeline is divided into separate frames, whose speed of playing is determined by the current tempo. The tempo can be changed at any frame. The timeline has a number of associated tracks, where, apart from the script track, any object can be placed on any track. 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 (although the position of an object over time is visualised much more clearly in the MET++ system). Sections of the timeline can be cut, copied and pasted. Jumps to other parts of the timeline are implemented via a "goto frame" command in the scripting language; each frame, or hotspot within a frame, can have an associated script.

The timeline gives a clear overview of the complete presentation. For long presentations, however, it is difficult to navigate around. Scene breaks can be deduced by sudden changes of objects on the timeline, but are not explicitly marked.

3.2.2. The Integrator

An explicit goal of the authors [35] is to create a development environment for interactive multimedia applications that relieves a developer of programming work. The central tool in the environment is the high-level Integrator which is used to assemble various media items into a multimedia application and to specify ways of interacting with the application.

The basic metaphor used in the Integrator is the musical score (similar in essence to the timeline in Fig. 3), where a pool of multimedia data items can be sequenced in virtual time, and then mapped for display onto real time. In the Integrator the multimedia presentation is represented as a set of horizontal tracks (analogous to staves in a score and similar to the CMIFed channels) (see Fig. 9). Timing and synchronization of multimedia items within a single track are determined by their horizontal positions on that track. Timing and synchronization of multimedia items across different tracks are determined by vertical relationships of objects across tracks (similar to synchronization arcs in CMIFed). As well as media item tracks the Integrator allows input or control tracks, and a timing track which allows the timing of the tracks to be altered (analogue of a baton in HyTime [20].

Authoring is carried out by placing icons representing the media items on one of the tracks at a specific time. A static item, such as an image, will remain on display until another item occurs on the same track. Composite objects can also be created, e.g. built up from an image with graphic overlay, or a slide show. A composite object can be placed on the control track of the timeline, and can be opened to view the layout of objects on its own timeline. The composite object appears as one object, which makes it difficult to get an overview of all the objects making up the presentation. Also, time dependencies between objects at different levels of the hierarchy are impossible to specify.

The authors observe that a timeline represents the parallel nature of multimedia applications better than a flowchart. They have, however, included several "flow" operations that can be added to the control track of the timeline, including iteration and conditionally branching constructs. These add to the power of the specification language, but make the visualisation of the presentation on the timeline difficult to interpret.

Transitions—visual effects when moving from one item to another, e.g. a video fading to black—can be specified in the system. These are attached to the media items rather than being separate objects themselves, and can occur at the beginning of an item, e.g. fade up from black; at the end of an object; or join two objects, e.g. a video dissolves to another video.

3.2.3. MAEstro

MAEstro[3] [8] is a suite of applications allowing the editing of multimedia documents. The timeline representation of a document has a track for each medium being used, e.g. compact disc audio or

3. MAEstro is now commercially available. For further information see `<URL:http://www.mae.com/>`.

laserdisc video. An author places each media item at its start time on the timeline. The duration of a continuous media item is determined by its media-dependent duration, whereas text is assigned a duration (by the system or the author). The multimedia composer can request the media players to select part of media item for display, so that individual data files do not need to be created for every media item used in the presentation.

3.3. Flowchart-based Authoring Systems

Authoring in the flowchart paradigm (illustrated in Fig. 3) is similar to programming the presentation in a procedural way, but with an interface improved by providing high-level icons for visualising the actions that take place. The icons are assembled in a structure which is visualised and manipulable by the author. Systems using this paradigm tend to provide more powerful facilities for expressing user interaction than solely structure or time-based systems.

3.3.1. Authorware

Authorware (chapter 12 of [4], [18], [21][4], [34], [36]) is a powerful and flexible commercial system for creating interactive multimedia presentations for computer based training and kiosk applications. To create a presentation, icons are selected from a palette and incorporated into a flowchart defining the sequence of events in the presentation. Flowcharts can be grouped into subroutines and nested to arbitrary levels. This is often necessary, since there is a limit to the display area for any one flowchart. The hierarchy of subroutines can be used by the author as a storyboard for working on the presentation top down—first by stating the sections in the presentation and then filling them in. The flowcharts remain procedural however, and there is no way of getting an overview (via a timeline) of which media objects will be played on the screen when. Interactions, on the other hand, can be fairly complex, and go far beyond “jump to here” commands.

3.3.2. IconAuthor

IconAuthor (chapter 12 of [4], [18], [21][4], [34]) is a sophisticated commercial package providing a suite of editors for different data types. Objects created by these, and other external editors, can be included in the end presentation, built in the central application builder. This is icon-based with flowcharts constructed from a library of icons representing actions comparable to those found in conventional programming languages and more specialist icons for media-presentation and interaction. There is no enforcement of any programming discipline, so that large, unstructured graphs can be created. The author is given some help with flowchart navigation through being able to zoom in and out of the flowchart representation, and being able to simplify the display by collapsing or expanding collections of icons. Previewing the presentation is possible from the beginning or from a selected starting point.

3.3.3. Eventor

The creators of Eventor (Event Editor), [9], argue that authoring facilities should apply a divide and conquer approach, which they support by providing three different views of the presentation—temporal synchronizer, spatial synchronizer and user interaction builder. They distinguish time-based systems (in our terminology, timeline based) and event-based systems (analogous to our flowchart paradigm), and aim to incorporate the advantages of both in an authoring system. Eventor is based on CCS, Calculus of Communicating Systems, a formal specification mechanism. This allows a formal specification of the behaviour of the presentation, which can be used for checking, for example,

---

4. The packages Authorware and IconAuthor were compared in [21] in 1993. Note, however, that both packages have newer versions on the market (Authorware 3.0 and IconAuthor 5.1) which may differ substantially from those reviewed.

syntactic correctness. In addition, they provide automatic aids for validating, for example, temporal constraints.

Basic units of programming in Eventor are media items (called basic objects). Temporal synchronization can specified among basic and composite objects (a collection of basic objects). Synchronization points (similar to anchors in the AHM) can be marked in video. The temporal synchronizer visualises the composite objects in a flowchart style structure. The spatial synchronizer allows the author to specify paths and scaling transformations by demonstration—these are tightly coupled to the temporal synchronizations. (While this provides a more direct method of interaction than in MET^{++}, section 3.1.3, the latter visualises the spatial movements in time more explicitly, Fig. 7.)

3.4. Script-based Authoring Systems

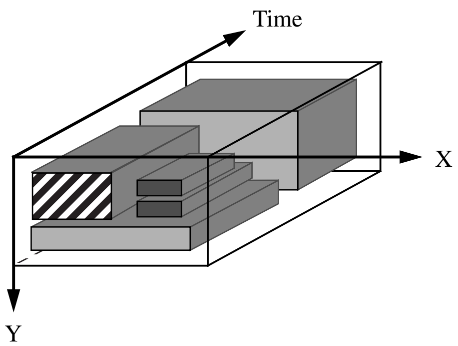
A script-based system gives an author a language for specifying the components, their timing, layout and interactions within a presentation. With no further support, this is a tedious, low-level method for specifying a multimedia presentation. It can, however, be the most flexible, since with a rich language the author is not constrained to the document model implemented in the system.

3.4.1. Videobook

The Videobook system, [29], was designed to incorporate a time-based, media-composite data sequence with the hypertext node and link concept, allowing the construction of composite multimedia nodes. The system presents media items and buttons (sources of links) according to a script specifying their presentation parameters—timing and layout. The script is visualised as a three-dimensional display showing the layout of each object along a timeline (Fig. 9). The system thus provides a low-level scripting language for the author to specify a presentation, which is then given a higher-level visualization along a timeline. The author is provided with some amount of structuring support, since each scene is defined in a separate script and scripts for scenes can contain nested sub-scenes. Synchronization of objects is specified by giving the start time of an object with respect to the scene. The multimedia presentation does have an underlying structure-based paradigm, but this is interpreted from the author-defined script, rather than being the basis of editing and used for generating the script.

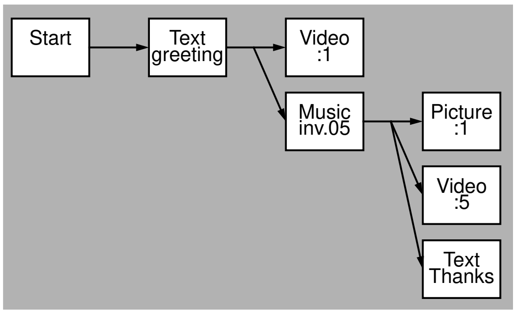
3.4.2. Harmony

Harmony, [10], has goals similar to those of Videobook, integrating continuous media items into a hypertext system. Each object is considered a node and there are links between nodes. Links are used for expressing the timing relations between nodes, using expressions such as



The layout and start times ofthe media items, defined by the author in a script, are given a three-dimensional visualisation. (Copied without permission from Fig. 3, [30].)

Figure 9. *Videobook* scene.



The presentation structure as shown in the Harmony scenario viewer. (Copied without permission from Fig. 5, [10].)

Figure 10. *Harmony* scenario structure.

`<aVideo, started: 30, aMusic, play>` which specifies that the piece of music starts 30 seconds after the video finishes. The notion of an object group is introduced, where, if an object group is the destination of a link, a message is broadcasted to all members of the group when the link is traversed. A scenario viewer displays the structure of a scenario. This is illustrated in Fig. 10, showing the time ordering or relations among media items.

4. Analysis

While the main goal of the authoring process is the expression of the author's intended narrative, specifying the constraints defining the display of the multimedia items composing the presentation often overwhelms the production process. It is important, then, that an authoring environment provides assistance in managing these constraints as well as assisting in the expression of the narrative. From the support provided by the systems discussed in the previous section we can identify three types of high-level constraints that need to be specified as part of the production process:

- *Logical, or structural, constraints:* The narrative as a whole can be broken down into sections and each section created independently of other sections. Media items that together express an idea or a concept can be grouped together and the group re-used elsewhere in the presentation. Connections, or logical relations, among related groupings can be created. (In the case of hypermedia these can be used directly for navigating through the information—an interaction constraint.)
- *Layout constraints:* Within a logical grouping, temporal and spatial layout constraints need to be specified. These may be relative to a timeline (in the temporal case) or a window (in the spatial case), or may be defined as relations between the constituent objects. Also, behaviours of objects as they are displayed, replaced or removed can be described (e.g. transitions).
- *Interaction constraints:* The presentation can be constrained by user interactions that influence the order in which information is displayed. In Fig. 1, for example, an author has specified boxed items that are linked to other parts of the presentation; these allow a user to navigate through the information. Other forms of interaction include selecting an answer from a multiple choice question, or typing in an answer to a question.

These constraints need to be specified in terms of a document model which can be expressed to the author, perhaps in several ways, and manipulated. In order to reduce the complexity of the overall task, it can be simplified by partitioning it into subtasks and manipulating each element separately, where appropriate using different tools. For example, in the world of choreography [4], multiple representations of the dance can be provided—stage, sequence editor, and timeline. A number of the systems described in the previous section use multiple representations each with their own editor, e.g. CMIFed section 3.1.1, Athena Muse section 3.1.2, Integrator section 3.2.2, and Eventor section 3.3.3. Other, more low-level constraints, concerning data formats are also discussed.

4.1. Structure

A number of authors, [30], point out the need for giving access to the structure of a presentation, or, in other words [21], the conceptual and perceptual objects and relationships the author is concerned with, although others argue for keeping composition hidden [19]. We take the view that structure exists implicitly in many creative works and, where it does, it should be made explicit and editable.

4.1.1. Explicit representation and manipulation of structure

Authors perceive structure in a presentation and they work at different levels within this structure. An authoring system should show the structure as close as possible to the way the author perceives it, and allow the author to move around and manipulate it while maintaining the context of the current detail of working.

For example, although film is often seen as a linear medium, it also has its own, normally implicit, structure. Nearly any film (regardless of content or genre), can be broken up into parts which correspond approximately to paragraphs, sections and chapters [32]. Each of these parts can be manipulated as a whole, or at the scene, sequence or shot level [23]. In editing a movie, it may be necessary to "zoom-in" on a small temporal chunk — say a second's worth of video — in order to analyze subtle effects of a scene transition. At other times, it may be necessary to switch to a high level overview of the movie in order to navigate to a neighboring scene [27]. This multi-level perspective also exists at the application level. In terms of our dance example, a choreographer needs to move between the design of the overall dance and the design of the more detailed levels of section, phrase and particular movement or gesture [4]; in the same way, a multimedia designer will need to focus in on a small detail section, or to work at creating over-all relationships in the document.

To ease the authoring burden a system should, where possible, be able to deduce low-level details from high-level descriptions supplied by the author. For example when the author groups items to be displayed at the same time they should start and finish simultaneously by default, and allow the author to specify overrides as required. Examples of deriving timing from structural constraints can be seen in CMIFed section 3.1.1, MET^{++} section 3.1.3, and Mbuild section 3.1.4.

Making the structure of a presentation explicit is not only useful for viewing and manipulating it, but also for previewing parts of the presentation. This allows the viewing of document changes within a local context, keeping overhead to a minimum, while still allowing the edits to be viewed in a larger context when necessary.

4.1.2. Providing support for top-down and bottom-up structuring

Just as computer programmers work in differing ways during the development of a large piece of software, multimedia authors vary in their approach to design. Some work top down, first creating a storyboard by specifying a collection of inter-item constraints and then filling in the details, while others work bottom up, first collecting/editing source items and then weaving a story around them. In reality, most authors use both approaches during the development of a single document.

For example, Mackay and Davenport [23], discuss sculpting a documentary while shooting and editing the video material. This requires a large amount of work editing and re-editing the material. Tools are thus needed for easy grouping of existing objects and filling in details of higher-level objects. In a similar way, choreographers [7] either begin with the broad spatial outlines and develop the detailed movements later, or start with some specific movement and go on to develop the phrases and sections.

A number of the systems described in this paper provide support for top-down and bottom-up structuring. CMIFed, using the hierarchy view described on page 8, and Mbuild, section 3.1.4, allow the creation of children of an existing composite object. CMIFed and MET^{++}, section 3.1.3, allow the creation of a composite structure around an already existing object (composite or media item).

4.2. Presentation Specifications

4.2.1. Timing constraints:

Temporal constraints can be defined in two fundamentally different ways—between an object and a timeline, or among (possibly composite) objects. The power of the latter approach is that when an author edits or replaces an object this does not require the modification of other objects or their position relative to a timeline. An analysis of specifying temporal constraints among objects is given in [3]. Here, constraints such as "*start simultaneous with*" are specified graphically, but are not translated to a timeline representation. This gives the author a powerful way of constructing complex timing relations, but does not give a visual overview of the final result. Timeline based systems provide an intuitive way of viewing the temporal relations in a presentation, but do not provide the power of describing relations between objects. A better solution it to combine the power of defining relations between objects with a timeline view derived from these relations. This approach is taken in the systems CMIFed (using the channel view described in section 3.1.1), MET++ (section 3.1.3), and Mbuild (section 3.1.4), which derive a time-based view from the structure of the presentation and relations defined among the media items.

4.2.2. Support for assigning layout

All the systems discussed allow an author to place media items on a screen. The placement of an item, however, is not normally regarded as specifying a "constraint", although there are examples of systems where this is made more explicit—MET^{++}, Fig. 7, visualizes the movement of an object in both vertical and horizontal directions; Director, section 3.2.1, allows an author to specify the position of an object through a number of frames by specifying a path (although the position can later be changed on any frame). There is, as is the case with timing constraints, a choice of whether spatial constraints are specified with respect to an external entity, in this case a window rather than a timeline, or with respect to the spatial position of other objects. In the systems covered in this paper, however, no mention is made of the latter case, which has as a consequence that spatial layout is not derived from the structure of a presentation. (The former may be an indication that solving the constraints in two spatial dimensions is more complex than the one-dimensional case of the timeline. On the other hand, authors may not need the power of specifying constraints among objects within the spatial dimensions.)

High-level support is provided for object layout in CMIFed, where static channels (section 3.1.1) define areas of a window where screen-based media items can be displayed. Using combinations of channels, an author can create a consistent look and feel in sections of the presentation. A more dynamic use of channels would be to allow an object displayed within a channel to follow a predetermined path, or to allow the channels themselves to follow a path (or even both simultaneously!).

4.2.3. Transition effects

As well as presentation specifications for media items themselves, properties of transitions from one media item to another, or groups of items (scenes) can be specified. These define effects such as fade out to black, wipe left to right, dissolve to next scene, etc. In the Integrator, section 3.2.2, a transition effect is specified as a property of the beginning or end of an object. A "dissolve" can be created by having the first object fade out to black and the new object fade in from black. Transitions need not be limited to media items, but can be specified as properties of composites, for example scenes.

In particular, in the AHM, transitions could be specified as presentation properties of links, so that when a link is followed the source context can, for example, fade out and the destination context fade in, resulting in a dissolve to the new scene. In none of the systems discussed can transitions between scenes be specified.

Although transitions are normally experienced as temporal transformations, they can also be defined spatially (chapter 7 of [4]), e.g. two overlapping images could blend into one another in their area of overlap.

4.3. End-User Interactions with Presentation

Document interaction features provide a mechanism for the end-user to interact with the final presentation. The concern here is "how does the document provide a control interface between the author and the user." Three sets of document interaction features can be identified:

- *Navigation:* Information is normally presented in a continuous manner, conforming to the timing specifications in the document. In addition, a number of the systems allow the author to specify a set of choices that enable the reader to select one of several follow-up locations. This is generally done by creating an active area on the screen and attaching destination information, via either a link object or a script. This type of feature requires support for the selection of such active regions and the ability to specify target locations. One form of this interactive navigation is specified through links, anchors and contexts as described in the AHM, section 2.1.
- *Presentation control:* Interactive navigation implies the presence of choice points at specific places in a document. Even where there is no choice of destination for the reader, some systems allow the presentation to halt (for example via a "pause" script) and then wait for a signal from the reader that they are ready to continue. An alternative is to give the reader a way of halting the presentation at any point. Another form of control that could be provided is of the speed of presentation, and whether it should run backwards or forwards. In the MET^{++} system, [1], the user is even able to record an event.
- *Environment control:* Little support is given for a reader to tailor the environment in which a document is presented. Support could be provided for varying the parameters of a document presentation. Examples in the CMIFed player are: a reader can select which channels to play, enabling, e.g., a language to be selected; the window(s) containing the presentation can be resized.
- *Application interactions:* Most of the features summarized so far deal with an author's ability to create multimedia presentations and a user's (restricted) ability to interact with them. A final class of interactions is between the user and the underlying application. This area can be very broad, since it is limited only by the nature of the application. As an example, consider applications designed for use in an interactive learning environment. Here, support is often required for the notion of student tracking and testing. Systems requiring a high degree of application-specific interaction make use of specialized suites of tools in addition to more media-specific authoring tools. The icon-based languages in systems such as Authorware, section 3.3.1, and IconAuthor, section 3.3.2, are examples of the flexibility that needs to be provided for creating this type of specialist interaction.

4.4. Data Object Manipulation

Object interactions allow an author to manipulate source media items in a document. The degree to which objects can be manipulated varies widely. A representative sampling of options is:

- *Data object editing*: An authoring system could import and/or create items in one or more data formats for each medium (including text, bitmaps, graphics, audio, video, animation). Often, an authoring system will use a set of limited internal editors to create, edit and manipulate items. In general, a more powerful approach for the manipulation of data is to provide access to good external tools (at least one for each data type), rather than being constrained to use mediocre internal tools. An exception is often made for text, since, in spite of the high degree of sophistication of external text formatters, the provision of simple, direct editing tools for text is perceived as important.

- *Data object scaling*: It is typically useful to support some form of object scaling that allows the spatial, or temporal, dimensions of source data to be altered for a particular presentation. Having created a picture, for example, it is useful to be able to scale it to fit the space available; similarly an animation could alter its speed to fit in with a piece of music.
  A more challenging problem is the generalized scaling of information, in which all types of data can be scaled to meet particular platform constraints. Examples of this may be the conversion of 24-bit images to 8-bit images, or the reduction of high quality sound data for transmission across limited bandwidth. Although scaling is a useful operation, because of the high processing power required even simple implementations are largely unsupported by current systems.

4.5. Facilities for Structured Multimedia Authoring

Given the breadth of experience reported we can construct a robust list of system requirements for supporting the authoring of interactive multimedia. The most important authoring requirement is that the narrative structure of a presentation be visible and editable by the author. This should be used a much as possible for generating other detailed aspects of the presentation, in particular timing constraints. Where further specifications are needed, these should also be given in terms of constraints and used to generate a time-based representation. The timeline representation should also be editable, for example, for changing the duration of objects or timing constraints between objects. Reusable high-level presentation specifications should be available for the definition of, e.g. reusable window areas and associated attributes (such as background colour), or media-dependent attributes such as font styles. The editing environment should be able to display media items created in specialist media editors. Where only simple interaction with the reader is required, easy to use tools should be provided (e.g. using a hypermedia link editor to define choice points). When interaction is more complex, tools more akin to standard programming languages are required.

We have introduced CMIFed as an example authoring environment, designed to provide support to the author for creating presentations. While the current prototype has proved an extremely useful testbed for our ideas, it is by no means complete. Based on the previous discussions, we list here some extensions we believe would be improvements to the system as it currently stands, thus providing a more rounded example of an integrated multimedia authoring environment.

The timeline representation of the presentation in the channel view has no means of zooming in and out. For longer sequences it would be desirable to have some means of magnifying parts of the timeline.

Timing relations between media items can be defined as synchronization arcs in the channel view. There are three restrictions in the current implementation. Anchors are not supported in dynamic media, so that instead of, for example, marking a word in the sound file and being able to synchronize a subtitle with it, the author has to estimate a time delay, and run through the presentation a number of times to fine tune it. Synchronization arcs are defined only as delays and have no notion of how strict the delay should be treated, e.g. is a delay specified as 3.5 seconds to be interpreted as 3.5 +/- 0.5 seconds or 3.5 +/- 0.1 seconds. A third restriction comes about through the current user interface. Since structure is shown only in the hierarchy view, and since synchronization arcs can be defined only in the channel view, timing constraints can be specified only between two media items and not between groups of media items.

CMIFed does not support the definition or playing of transitions. These could be added to a continuous presentation in the hierarchy view (between serial structures) and displayed in the channel view. Adding transitions to links could be done by specifying the transition type for the source and destination contexts associated with a link, thus allowing different transitions for the same objects used with different links. Alternatively, transitions could be attributes of the (composite) objects themselves, but then the transition type could not be varied per link.

At a per channel level there are three new facilities which could be provided: cropping facilities for images and video could be provided (scaling of images is already provided), which would allow part of an image to be selected for display in the channel; animation of objects within a channel would be useful for creating simple animations; assigning a picture as a background "colour" would allow for more professional-looking presentations, but requires that a transparent background "colour" be implemented for the other channels.

The player should provide a control for playing the presentation at different speeds and in different directions, such as fast forwarding or slow motion.

Lastly, CMIFed has been designed as a general-purpose document authoring facility without support for special-purpose application features. Examples of useful features are support for usage statistics, or student tracking and testing in learning applications. This type of support is difficult to provide within a specific document model so that special-purpose tools need to be written. We have extended the system to support an interactive multimedia interface to a simulated business game, [13], which led to the incorporation of a special channel type for communicating with external processes. The structure of the multimedia document itself remains the same.

5. Conclusions and Future Directions

None of the systems described in this STAR report has all the recommended features of a powerful but easy to use multimedia authoring environment. If such a system were created it would be a complex environment giving different ways of viewing and manipulating aspects of a presentation, but the author would still be required to specify most of the implementation details explicitly. It is clear, however, that an author wants to concentrate the narrative rather than the constraints among the items when creating a presentation. Ideally, the author, or indeed an end-user, would need to give only an indication of what the presentation should be about and the system would construct a coherent, flowing presentation. In order to attain this level of automation we need rich semantic-based descriptions of the media items available—[2], [6]. These descriptions currently require a large amount of time and effort to produce. MacNeil, [24], has investigated generating presentations on the basis of authored examples. He splits the problem into logical, spatial and temporal perspectives. Work on each of these aspects could be done separately, for example we have already presented work on generating timing constraints from higher-level specifications: Mbuild section 3.1.4, CMIFed section 3.1.1and [3].

A different approach is to simplify the creation process by deducing as much as possible from the task the author is carrying out. For example, when the author assigns a data object to a structure element in CMIFed's hierarchy view, the system could deduce a number of things rather than have the author define them. Instead of requiring that the author assign a channel for each data item, the system could make a best guess by assigning an image to the only image channel in the current layout, or by presenting a choice of image channels if there were more than one. A number of such "first-step" automations would provide insights into the complexity of the different aspects of the task.

However easy to use or complex authoring systems become, there will remain a conflict between regarding a multimedia presentation as a document created with specialised editing tools, or as a pre-programmed sequence of events and interactions. The former allows the creation of easy-to-use specialist tools based on a document model; documents so created can be transported among systems that can interpret the model. The latter remains more akin to programming, allowing more flexibility of interaction to be specified, but only in the hands of an expert user. Current commercial authoring systems tend towards the flexible, complex side, where users are expected to spend time learning the system.

5.1. Possible, Desirable and Probable Futures for Authoring Systems

As a last comment, we look at different future scenarios for the development of multimedia authoring systems.

- *Possible:* easy to use structured authoring systems, with derived timelines, will become more of the norm than the exception. Commercial systems will gradually take on new features as the market becomes more experienced in multimedia authoring and begins to demand more from the systems.
  Documents that have been created can be stored in terms of a specific HyTime [20] DTD (Document Type Definition, [33]) and read in by systems that can interpret that DTD.
- *Desirable:* even more support will be provided by an authoring system, for example deriving structure from collections of related media elements, and deriving timing and layout relations.
  Just as the specification language HTML (HyperText Markup Language) has been developed for the WWW (World Wide Web), a HyTime based language at least as powerful as the Amsterdam Hypermedia Model will be developed for the web. Different viewers will be written that can interpret this, in the same way that different HTML viewers have already been developed.
- *Probable:* commercial interests will ensure that each authoring package uses its own format, and separate conversion programs may be written to convert one popular format to another.
  HyTime will be perceived as too inflexible by multimedia designers, and too complex to implement by systems designers.

Acknowledgements

Guido van Rossum, Jack Jansen and Sjoerd Mullender designed and implemented the authoring environment CMIFed.

References

1 P. Ackermann (1994). Direct Manipulation of Temporal Structures in a Multimedia Application Framework. In Proceedings: *Multimedia '94*, San Francisco, CA, Oct, 51 - 58.

2 T. G. Aguierre Smith (1991). Parsing movies in context. In Proceedings: *USENIX* Summer, Nashville, TN, 157 - 167.

3 M. C. Buchanan and P. T. Zellweger (1993). Automatic temporal layout mechanisms. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug, 341-350.

4 J.F. Koegel Buford (ed.) (1994). Multimedia Systems. Addison-Wesley, New York, New York. ISBN 0-201-53258-1.

5 Dick C.A. Bulterman, Guido van Rossum and Robert van Liere (1991). A Structure for Transportable, Dynamic Multimedia Documents. In Proceedings of the Summer *USENIX* Conference, Nashville, Tennessee, 137-155

6 V.A. Burrill, T. Kirste & J.M. Weiss (1994). Time-varying sensitive regions in dynamic multimedia objects: a pragmatic approach to content-based retrieval from video. *Information and Software Technology* Journal Special Issue on Multimedia 36(4), Butterworth-Heinemann, April, 213 - 224.

7 T.W. Calvert, A. Bruderlin, S. Mah, T. Schiphorst and C. Welman (1993). The evolution of an interface for choreographers. In Proceedings: *InterCHI '93,* Amsterdam, Apr, 115 - 122.

8 G.D. Drapeau and H. Greenfield (1991). MAEstro — A Distributed Multimedia Authoring Environment. In Proceedings of the Summer 1991 *USENIX* Conference, Nashville, TN, 315 - 328.

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

10 K. Fujikawa, S. Shimojo, T. Matsuura, S. Nishio and H. Miyahara (1991). Multimedia Presentation System 'Harmony' with Temporal and Active Media. In: Proc.s of the Summer 1991 *USENIX* Conference, June, Nashville, TN, 75 - 93.

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

12 R. Hamakawa and J. Rekimoto (1994). Object composition and playback models for handling multimedia data. *Multimedia Systems* 2: 26 - 35.

13 L. Hardman, G. van Rossum and A. van Bolhuis (1994). An Interactive Multimedia Management Game. To be published in *Intelligent Systems*.

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

15 L. Hardman, D.C.A. Bulterman and G. van Rossum (1993). Links in hypermedia: The Requirement for context. In Proceedings: *ACM Hypertext '93*, Seattle WA, Nov, 183 - 191.

16 L. Hardman, G. van Rossum and D.C.A. Bulterman (1993). Structured multimedia authoring. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug, 283 - 289.

17 M.E. Hodges, R.M. Sasnett and M.S. Ackerman (1989). A construction set for multimedia applications. *IEEE Software,* 6(1), Jan, 37 - 43.

18 R. Howorth, P. Monckton, S. Parnell, I. Burley, L. Cole and R. Schifreen (1994). Stop, Look, Listen. *PC Magazine*, August, 151 - 199.

19 S.E. Hudson and C.-N. Hsi (1994). The Walk-Through Approach to Authoring Multimedia Documents. In Proceedings: *Multimedia '94*, San Francisco, CA, Oct, 173 - 180.

20 HyTime. Hypermedia/Time-based structuring language. ISO/IEC 10744:1992.

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

22 G. Liestøl (1994). Aesthetic and Rhetorical Aspects of Linking Video in Hypermedia. In proceedings of (Sixth ACM Conference on Hypertext) *ECHT '94*, 217 - 223

23 W. E Mackay and G. Davenport (1989). Virtual video editing in interactive multimedia applications. *Communications of the ACM*, 32(7), July, 802 - 810.

24 R. Macneil (1991). Generating multimedia presentations automatically using TYRO, the constraint, case-based designer's apprentice. In Proceedings: *IEEE 1991 Visual Language Workshop*, 74 - 79.

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

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

27 M. Mills, J. Cohen and Y. Y. Wong (1992). A magnifier tool for video data. In Proceedings: *CHI '92*, May, Monterey CA, 93 - 98.

28 S.R. Newcomb, N.A. Kipp and V.T. Newcomb (1991). 'HyTime' the hypermedia/time-based document structuring language. *Communications of the ACM*, 34(11), Nov, 67 - 83.

29 R. Ogawa, H. Harada and A. Kaneko (1990). Scenario-based hypermedia: A model and a system. In Proceedings: *ECHT '90* (First European Conference on Hypertext), Nov, INRIA France, 38 - 51.

30 R. Ogawa, E. Tanaka, D. Taguchi and K. Harada (1992). Design Strategies for Scenario-based Hypermedia: Description of its Structure, Dynamics, and Style. In Proceedings: *ECHT '92* (Fourth ACM Conference on Hypertext), Nov, Milano, Italy, 71 - 80.

31 G. van Rossum, J. Jansen, K.S. Mullender and D.C.A. Bulterman (1993). CMIFed: a presentation environment for portable hypermedia documents. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug, 183 - 188.

32 B. Rubin and G. Davenport (1989). Structured content modeling for cinematic information. *SIGCHI Bulletin*, Oct, 21(2), 78 - 79.

33 SGML. Standard Generalized Markup Language. ISO 8879:1986. Amendment 1: ISO 8879:1992/AM1:1992.

34 J. W. Semich (1992). Multimedia tools for development pros. *Datamation*, Aug 15, 90 - 95.

35 A. Siochi, E.A. Fox, D. Hix, E.E. Schwartz, A. Narasimhan, W. Wake (1991). The Integrator: A Prototype for Flexible Development of Interactive Digital Multimedia Applications. *Interactive Multimedia* 2(3), 5 - 26.

36 N. West (1993). Multimedia Masters: A guide to the pros and cons of seven powerful authoring programs. *MacWorld*, Mar, 114 - 117.