CWI Quarterly Volume 7 (1) 1994, pp. 47 – 66

Authoring Interactive Multimedia: Problems and Prospects

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

The creation of a multimedia presentation is a non-trivial task. It involves skills that are not readily available to users and it requires support not generally available from authoring software. In order to understand the basic problems of multimedia authoring, this article considers the requirements for defining interactive, dynamic presentations. When contrasted against the facilities available in current-generation commercial authoring systems, we can see that their focus is often on low-level details rather than high-level structure. The prospects for future editing systems are somewhat brighter: support for high level editing can be provided. As an example, we describe the CMIFed authoring environment; CMIFed not only supports authoring at a high level but also incorporates most low-level features found in current systems.

1. Introduction

The rapid development of computer-based input/output technology has forced significant changes in the way that information is encoded and exchanged in our society. The development of the laser printer, for example, served as the catalyst for a wide range of electronic publishing software–software that allows relatively untrained users to approximate the skills of typesetters, draftsmen and graphic artists. In the same way, the integration of loudspeakers, microphones and video interfaces into moderate-cost computers allows the creation of electronic presentations that until recently would have required the use of professional production facilities. While this new technology enhances the expressive power of computer-based presentations, it also results in the further integration of the once-separate tasks of content creation and presentation definition: just as the development of electronic publishing forced writers to become typists and editors, the emergence of multimedia forces authors to become actors, producers and directors. This places a significant burden on the tools required to support creators and developers of multimedia presentations.

The success of electronic publishing is due, in part, to an accepted model of how information can effectively be presented on a two-dimensional printed page and to the fact that general population literacy means users are trained in the production of source material; while specific techniques for typing and editing need to be learned, most users already know what a document is and how to read and write. Unfortunately, the production of (effective) multimedia presentations requires a new set of skills that are not readily available to many potential users. Aside from the obvious problems of creating audio and video source material, most users have no experience with composing complex interactions of time-based data under the constraints of a particular computer system. Where electronic publishing has been able to draw on centuries of experience that collectively defines how books, newspapers, letters and informal notes are structured, the process of multimedia authoring has only a few decades worth of examples to guide the development of presentations, and most of these examples–such as television and radio productions–involve artistic and technical expertise that is not widely understood.

This paper outlines problems that are particular to the development of multimedia presentations. We provide a review of the fundamental characteristics of the multimedia authoring process and attempt to abstract the essential features required by authoring tools. This discussion includes a review of the features found in common commercial systems and a discussion of the basic facilities required to manage the often complex task of integrating several time-based information streams in a single presentation. We then present a prototype system that provides an author with a structured, multi-level view of the presentation under construction. The main advantage of our approach is that it provides an authoring model that separates manipulation of structural elements of a presentation from those that define content. It also makes use of a flexible presentation model that makes migration of a document to a wide range of target platforms possible.

In the following sections, we provide an analysis of the requirements for multimedia authoring and then describe the authoring facilities available with CMIFed (CWI Multimedia Interchange Format editor), a structured multimedia authoring system developed at CWI. We preface our discussion with a review of the general characteristics of interactive multimedia presentations.

2. Example 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 Figure 1, which illustrates three fragments from a tour of the city of Amsterdam. In the top fragment, which is analogous to

FIGURE 1. An example interactive multimedia presentation

a presentation 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 such alternative 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 loud-speakers.
The application illustrated by Figure 1 is representative of those found in information kiosks or electronic travel brochures. Other example application areas are: news and information servers, electronic books and, of course, entertainment. Note that commercial multimedia presentations are generally restricted to educational material and arts/entertainment; Table 1 summarizes the classification of 23 high-quality multimedia CD-ROM applications [2, 15, 18].

Most multimedia applications internally structure information as *documents*. Documents can be opened and read interactively or they can be displayed in the manner of a film: a fixed sequence of information fragments that are projected to a passive user. The multimedia document is a broad and general model for combining various types of multimedia information. (It is not the only model however; see the article on MADE in this issue for a more media/data oriented approach [11].)

One characteristic of document-based multimedia is the requirement for some form of interaction between the presentation and the user. If an interactive interface is provided–and if the underlying document model supports it–presentations can be structured as *hypermedia documents*. In a hypermedia document, the end-user can select a navigation path through the information via *links.* The user is typically constrained to following author-defined links, but the on-demand generation of links is a natural extension that is already supported in some text-only hyper-documents [5].

3. Approaches to authoring multimedia documents

The essence of multimedia authoring is that the author wants to communicate a message to the reader and so needs to control the timing and placement of objects in the presentation. As was discussed above, authoring multimedia is not unlike 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 [4]. 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.

While the main goal of the authoring process is the creation of an effective narrative, the process of managing multimedia information constraints often overwhelms the production of the presentation. It is important, then, that an authoring environment provides assistance in managing these constraints as well as assisting in the expression of the narrative. Three types of low-level constraints can be identified:

(1) *Intra-item constraints*: The presentation of each individual item is constrained by the characteristics of that item's data representation; video

TABLE 1. Example commercially available multimedia presentations

information, for example, has a particular frame rate, audio information has specific frequency ranges and sample sizes, pictures have dimensions and color-map information, etc.

(2) *Inter-item constraints*: The composite presentation is constrained by various logical and physical synchronization relationships *among* the constituent items; constraints may be based on representation (for example, making sure that a caption fits under a picture on a screen or that an audio and video sequence end together) or on content (where, for example, two items that provide related information are presented together). An analysis of different types of temporal constraints is given in [3].

(3) *Interaction constraints*: The presentation can also be constrained by user interactions that influence the order in which information is displayed. In Figure 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.

We first consider high-level authoring requirements, derived from the experiences of creators of dynamic and interactive presentations. We then look at low-level authoring requirements derived from the facilities provided in a range of commercially available multimedia authoring systems.

3.1. Authoring system requirements

At a high level of abstraction, three different requirements can be identified for assisting in the constrained control and placement of multimedia objects. Each of these is discussed in the following sections.

3.1.1. Supporting a multi-level document manipulation model

The development of a single document is rarely a straightforward matter of selecting items and arranging them as part of a narrative. As a result, it is usually desirable to allow authors to manipulate all three types of constraints concurrently (either from within one view or across related views.) Authors perceive structure in a presentation and they work at different levels within this structure. An authoring system should show the structure 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 [21]. Each of these parts can be manipulated as a whole, or at the scene, sequence or shot level [13]. 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 [16].

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.

An early example of an authoring system that partially supported this type of multi-level editing used a visual interface to creating hypermedia scenes. A three-dimensional "time-line" was developed, where the layout of the presentation could be seen along a time axis. Manipulation can take place at the object and scene levels [19].

3.1.2. Providing support for top-down and bottom-up construction

Just as computer programmers work in different 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 often use both approaches simultaneously during the development of a single document.

For example, MACKAY and DAVENPORT [13], 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 [4] 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.

3.1.3. Partitioning tasks to manage complexity

Even with support for multi-level work and top-down/bottom-up design, it is important that the various parts of the over-all development process be segmented into their disjoint components. Often, the data elements that make up each document can be viewed from three different perspectives: logical, spatial and temporal [14]. It is important to be able to manipulate each of these elements separately, perhaps using separate tools.

For example, in a system for producing educational documents, it was found that four distinct representational approaches were needed: directed graphs, multidimensional spatial frameworks, declarative constraints, and a procedural language. The authoring environment needed to support all representations to be useful [12]. Again in the world of choreography [4], the complexity of the task was reduced by using multiple representations-stage, sequence editor, and timeline.

The role of the authoring tool is also to manage communication among these different representation editors; for example, when an object is being played in a runtime environment it should be easy to find it in the different authoring views.

3.1.4. Providing a platform independent document specification

A final principle guiding the development of authoring systems is portability. Once a presentation has been completed it should be able to run on a number of platforms with no, or few, further modifications. Authoring complex presentations, just as writing complex code, is a task that should be designed to be portable-even across presentation platform facilities. 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. In earlier work, we developed the Amsterdam Hypermedia Model [8] which describes the information elements necessary for capturing the essence of an interactive multimedia presentation. This description can be expressed in different intermediate forms suitable for multiple presentation environments. (An example intermediate form could be HyTime [17].) We use the Amsterdam Hypermedia Model as a base on which we have built our authoring tools.

3.1.5. Reducing authoring effort

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. Another useful feature is to support the quick viewing of small document changes within a local context, while still allowing the changes to be viewed in a larger context where necessary. Both of these features reduce author effort and increase author productivity.

3.2. Current commercial multimedia authoring systems

A number of commercial authoring systems have been reviewed in [22], [23], [24] and [25].[1] Using these reviews as a base, we can distinguish four classes of authoring features that are available in at least some current commercial systems. The four classes of features are: author interaction, document interaction, data object interaction and application interaction features. A list of the classes and the supported features is given in Table 2.

[1]Systems described in these articles: *Act III*, Informatics Group; *Action 1.0*, Macromedia; *Adobe Premiere 1.0*, Adobe Systems; *Aldus SuperCard 1.6*, Aldus Corporation; *Animation Works Interactive 1.1*, Gold Disk; *Ask*Me Pro 2.12*, Ask*Me Information Center; *Authorware Professional 1.7.1*, Macromedia; *AVC 1.05*, IBM; *Challenger!*, SASI Software; *Cinemation 1.0*, Vividus; *Course Builder 4.0*, Discovery Systems International; *Grasp 4.0*, Paul Mace Software; *HSC InterActive 1.0*, HSC Software; *HyperCard 2.1*, Claris Corporation; *HyperWriter! 3.0*, Ntergaid; *IconAuthor 4.1*, AimTech; *Linx Industrial*, Warren-Forthought; *LinkWay Live!*, IBM; *MacroMind Director 3.1*, Macromedia; *Media Master*, Vision Imaging; *MediaMerge 1.0*, ATI Technologies; *MovieWorks 1.0*, Interactive Solutions; *Multimedia Desktop*, Datalus; *Multimedia Toolbook 1.5*, Asymetrix; *Overtext,* Probyte; *Passport Producer 1.0*, Passport Designs; *Quest 4.0*, Allen Communication; *Special Delivery 1.0*, Interactive Media Corporation; *Spinnaker Plus 2.1*, Spinnaker Software; *Storyboard Live!*, IBM; *TEMPRA SHOW*, Mathematica; *Test Factory 2.1*, Warren-Forthought; *TMM Producer Station*, TMM.

TABLE 2. Authoring system features

3.2.1. Author interaction features

Author interaction features allow an author to compose a particular presentation. While all of the facilities of an editing environment contribute to this task, we can identify the following collection of features as being central to the process of building a document:

- *Authoring paradigms*: A number of different styles are supported by commercial systems. The majority of these fall under one of three paradigms: *scripting*, *timeline* and *flowchart*. A graphical impression of these paradigms is given in Figure 2. 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 being less manageable in large presentations. It can also be difficult to integrate particular inter-item presentation constraints into a script. Timelines show the media items placed on different tracks along a time axis. These are useful for giving an overview of when which objects are displayed on the screen (and for how long), but not for manipulating presentations at a scene level. A flowchart gives the author a visual representation of the structure of the presentation. While systems using this approach are deemed simpler to use, they tend to show all the available structure, and–with one exception (Authorware Professional 1.7.1 [25])–there is no way of getting less-detailed views of a presentation. A fourth, but less common approach, is to the *slide*-based paradigm, where the presentation consists of a linear sequence of slides upon which a number of objects are placed (a slide can be thought of as one scene). In this case both the slides and the objects may have associated timing information.
- *Presentation (pre)viewing*: It is desirable while authoring to have some means of viewing the presentation as the reader will see it. This provides valuable feedback, much in the way that WYSIWYG text editors provide an author with an accurate picture of how a text document will look.

Some systems have separate authoring and viewing tools, forcing the author to jump between viewing the presentation and editing it. One system forces the author to compile the whole presentation into a specific format before it can be viewed, thus making a quick check of a small change extremely cumbersome.

- *Cross-platform development*: Given the arduous task of creating presentations, it is desirable to reuse a given presentation on as many different types of environments as possible. Several commercial systems provide this capability, but the migration is restricted to support for the Apple Macintosh and Microsoft Windows. No support is provided for migrating presentations between platforms with dissimilar hardware, or for systems where the presentation hardware may vary over time.

3.2.2. Document interactions

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 are found in commercial systems:

- *Interactive choice/navigation*: In most presentations, information is presented in a sequential order, based on the script, timeline or flowchart that defines 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, normally via a script. This type of feature requires support for the selection of such active regions and the ability to specify branch target locations in the script. A formal variant of interactive navigation can be found in hyper-documents [8]. None of the commercial systems reviewed supports a complete hyperinformation model.

- *Interactive presentation control*: Interactive navigation implies a pause-select-branch sequence at specific places in a document. Even where there are no destination choices for the reader, some systems allow the presentation to halt and wait for a signal from the reader that they are ready to continue. (An alternative is to allow the reader to halt the presentation at any point.) Another such tool that could be provided could control the speed of presentation.

- *Interactive 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 (such as the resizing of display areas or the selection of sound characteristics).

3.2.3. Data item manipulation

Data object interactions allow an author to manipulate source data items in

(a) Script (b) Timeline (c) Flowchart

(a) 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.
(b) A timeline representation gives the author a clear overview of which objects play when during the presentation. Manipulation takes place at the object rather than scene level.
(c) A flowchart shows the structure of the presentation clearly. The order of play can be deduced, but time is not represented explicitly.

FIGURE 2. Example authoring paradigms

a document. The degree to which items can be manipulated varies widely. A representative sampling of options is:

- *Data item editing*: Each 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 usually made for text; in spite of the high degree of sophistication of external text formatters, the provision of simple, direct editing tools for text is seen as important (since it takes place so often).
- *Data item scaling*: It is typically useful to support some form of object scaling that allows the spatial dimensions of source data to be altered to fit a particular platform. Having created a picture, for example, it is useful to be able to scale it to fit the space available for it in the presentation. A more challenging problem is the generalized scaling of information, in which all types of data can be scaled to meet particular presentation constraints. Examples of this may be the conversion of 24-bit images to 8-bit images, or high resolution sound data to low resolution objects. Although scaling is an intuitive operation, because of the high processing power required even simple scaling is largely unsupported by commercial systems.

- *Data item animation*: All systems allow an author to place objects on a screen. Some systems additionally allow the animation of objects via the specification of paths along which objects can travel.
- *Transition effects*: where animation refers to the behavior of a single object, transition effects provide support for moving from one *scene* to another, where each scene composed of a collection of objects. Different types of transition effects between scenes are often supplied, for example, wiping across screen, closing to a shape, or dissolving. Where these are available, systems supply in the order of 10 to 30 transition effects.

3.2.4. Application interactions

Most of the features summarized so far deal with an author's ability to create presentations and a user's (restricted) ability to manipulate 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. Such tracking/testing does not integrate seamlessly into a standard authoring model. In general, systems requiring a high degree of application-specific interaction make use of specialized suites of tools in addition to more conventional authoring tools.

3.3. Facilities for structured multimedia authoring

Our discussions on authoring tasks and commercial systems lead us to the following basic conclusion: an author of interactive multimedia presentations requires support for both high-level and low-level definition and editing tasks.

Given the breadth of experience reported we can construct a robust list of system requirements. The most important high-level authoring requirements are that construction of the narrative of the presentation be supported at multiple levels in a top-down or bottom-up manner, and that different representations of the underlying document be used to support different authoring tasks. In addition to the high-level support, low-level support is also essential. The core of this support should be directed toward inter-item constraints, since these are among the most complex in a document. (For example, tasks such as specifying that a sound fragment should start simultaneously with a picture and then continue playing until a later text item has completed are usually very difficult to express.) While intra-item constraints are not the primary focus of the authoring process, they are necessary for the smooth playback of the presentation and should be explicitly supported. Interaction with the end-user is less complex in the typical case of having only one end-user per active presentation, but still problematic in that the author can only propose, but not, predict a user's choices.

Although both sets of requirements need to be supported in a successful authoring system, commercial systems reviewed tend to be inadequate in providing high-level authoring support. Multiple levels of working are supported in only one flowchart- based system; only top-down construction is supported (most assume that a storyboard already exists); task partitioning has in some cases been taken so far that the different views no longer communicate with each other. For the lower level authoring tasks the reviewed systems perform much better. They provide a varied set of useful features which allow the author to describe intra-item and interaction constraints. There is less support for inter-item constraints, although transition effects can often be specified.

4. CMIFed–A multimedia authoring environment

CMIFed, (a multimedia authoring environment, has been designed to provide authors with a rich environment for high-level and low-level viewing and manipulation of a presentation. The three high-level authoring requirements stated above are met by partitioning the authoring tasks 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.

The structural information defined by the author in this 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 data nodes can be created. The *player* allows the author to see the presentation as the readers will see it.

Other requirements for an authoring environment, stated in Section 3.1, are also met by CMIFed: the presentation description is stored in a platform independent format (Section 3.1.4); it is possible to view the effects of changes either locally or more globally and the system deduces first-order timing details from the structural description (Section 3.1.5).

The hierarchy view, channel view and the player are described in the following sections. More detail on the interaction methods in each view can be found in [10].

4.1. The hierarchy view for structure manipulation

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

The hierarchy view (Figure 3 and Figure 4) is the primary authoring window, providing a means of displaying and manipulating the document describing a multimedia presentation. The document has a hierarchical structure whose leaf nodes are the data nodes which are played in the presentation, and whose non-leaf nodes are composite nodes containing a collection of other composite nodes and/or data nodes. The hierarchical structure is represented in the hierarchy view as an embedded block structure. Each data node 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 loud-speaker.

The structure of the Amsterdam tour is shown in Figure 3. The *Table of contents* corresponds to the screen in the upper part of Figure 1 and *Walking route* section contains the two clips shown in the lower part of Figure 1.

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.

FIGURE 3. Top level structure of the Amsterdam tour.

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 Figure 3, we see the structure shown in Figure 4.

Authoring work can be reduced through 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 Figure 4 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 Section 4.2). In Figure 4 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 burden on the author.

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 data node (a leaf node of the hierarchical structure).

FIGURE 4. Structure of the *Walking route sequence.*

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 multimedia presentation can be authored by first creating the structure and then assigning data nodes at the leaves of the structure. Alternatively already existing (data) nodes can be gathered together into higher levels of structure. A data node needs to be assigned to a channel (a logical output device) and to have a media object associated with it, usually by a reference to a file containing the data. When associating a file with a data node, 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.

4.2. 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 objects mapped onto the available logical resources (called *channels*). By supplying an extra layer above the physical resources, the channels, 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 objects to the available physical output devices.

The channel view shows the timing relations derived from the structure defined in the hierarchy view. The data nodes 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, Figure 4, and the data nodes in the channel view, Figure 5, is shown in Figure 6.

The diamonds at the top of the figure show the channel names (inactive channels are shaded). The data nodes 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.*

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 data node, for example, a short text string can be displayed in a larger font. The data nodes are each assigned to a channel.

As well as providing a device-independent description of a data node'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.

The ordering of data nodes 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 data nodes intersecting the dotted, horizontal line in Figure 5 are those presented during the gables clip shown in Figure 1.

While the majority of the timing relations among data nodes 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 objects. For example, the display of the text line of the *Gables2* subtitles in Figure 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 data nodes 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 prescheduling are also shown by highlighting the data nodes in the channel view. Further details on the scheduling are given in [20].

4.3. 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 hypertext model of anchor and link objects [8].

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. For example, in Figure 1 the *Walking routes* label takes the reader to a complete scene, with descriptions of the timing relations among the objects being played. It is not always the case that the whole scene needs to be replaced, for example, in the lower half of the figure the *Gables* label is a link to a smaller structure (the *Gables* box in Figure 4) and that the *Contents* text remains unaffected on the screen. When a link is followed there is a choice of whether the destination of the link replaces the scene that was being played, or whether it is displayed in addition. When there are active media playing in the source presentation there is a further choice as to whether the current scene should stop or continue playing when the link is followed. Support for these different options is provided within CMIFed. Adding navigation structures to multimedia presentations is discussed in greater detail in [9].

4.4. Player

The player interprets the system-independent specification of the multimedia presentation (in terms of the presentation's structure, logical resource allocation

(a) FIGURE 4 (b) FIGURE 5

Data nodes contained in the nested structure of the *Gables* section in (a) correspond to the data nodes enclosed by the stippled box in (b).

FIGURE 6. Correspondence between hierarchy and channel views.

and timing constraints) and plays the presentation on the available hardware. This process is described in detail in [20]. The player provides facilities, such as start, pause and stop, for the author or end user to control the playing of the presentation. Figure 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, e.g. to select one of a number of voice-overs in different languages. This is illustrated in Figure 1, where the reader has selected to listen to Dutch speech and read English subtitles. The corresponding channels can be seen in Figure 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 Figure 1) and the channels within will scale automatically (although we still require to implement a font-scaling algorithm for the text channels).

4.5. Possible enhancements to CMIFed

While we have satisfied the high-level authoring requirements for a multimedia authoring system and have included support for a number of low-level features, there are a number of low-level features which could be incorporated to improve

CMIFed.

CMIFed does not support the definition or use of transitions – a visual means of showing the movement from one scene to another. In a way similar to defining first-order and more detailed timing constraints, these could be added in the hierarchy view (between serial structures) and shown and edited in the channel view.

The player should provide a control for playing the presentation at different speeds, such as fast forwarding to a particular place or for playing a detail in slow motion.

The channel view has no means of zooming in and out (as the hierarchy view does). For longer sequences it would be desirable to have some means of magnifying parts of the channel view (similar to that described in [16]).

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); animation of objects within a channel would be useful for creating simple animations; assigning a picture as a background would allow for better looking presentations, but requires that a "transparent" background colour be implemented for the other channels.

Lastly, CMIFed has been designed as a general-purpose document authoring facility without support for special-purpose application features. Such features, such as supporting usage statistics or support for student tracking and testing in learning applications, provide useful features. It is still unclear if this type of support is possible within a general document model or whether special-purpose application drivers need to be written for each application areas. We are currently extending the system to support an interactive multimedia interface to a simulated business game [7]; we hope that this area will give us insights into the extensibility of the system in general.

5. Authoring prospects

Based on our own experience with using CMIFed and on the tasks described by other systems, it is clear that an author wants to concentrate on *what* rather than *how* when creating a presentation. Ideally, an end-user need give only an indication of his or her interests and the system will come up with a coherent, flowing presentation. In order to attain this level of automation we need rich content-based descriptions of the media items available–[1], [6]. These descriptions currently require a large amount of time and effort to produce. MacNeil [14], 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 work has been done on generating detailed timing constraints from higher-level specifications [3].

A different approach is to simplify the creation process by deducing as much as possible from a high level specification given by the author. An example from the CMIFed environment is that when the author assigns a data object to a structure element (in the hierarchy view), the system could deduce a number of things that the author needs to define. For example, instead of requiring that the author assign a channel for each data item, the system could make a best guess based on either the context of the placement or the contexts of the object. A number of such "first-step" automations would provide insights into the complexity of the different aspects of the task.

In spite of its minor shortcomings–and its prototype nature–the current CMIFed authoring environment demonstrates the usefulness of integrating high- and low-level authoring into a single environment. We feel that it provides a valuable testbed for constructing complex documents and for testing the limits of the document model.

ACKNOWLEDGEMENTS
The work reported in this paper was carried out by the Multimedia Kernel Systems group at CWI. Guido van Rossum, Jack Jansen and Sjoerd Mullender designed and implemented CMIFed.

REFERENCES

1. T.G. AGUIERRE SMITH (1991). Parsing movies in context. In Proceedings: *USENIX* Summer, Nashville, TN, 157 – 167.
2. A. AMIRREZVANI (1993). Great CD-ROMs for fun and profit. *PC World*, 239 – 248.
3. M. C. BUCHANAN and P.T. ZELLWEGER (1993). Automatic temporal layout mechanisms. In Proceedings: *ACM Multimedia '93*, Anaheim CA, 341 – 350.
4. 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, 115 – 122.
5. D.T. CHANG (1993). HieNet: A user-centered approach for automatic link generation. In Proceedings: *ACM Hypertext '93*, Seattle WA, 145 – 158.
6. G. DAVENPORT, T.G. AGUIRRE SMITH and *N. Pincever* (1991). Cinematic Primitives for Multimedia. *IEEE Computer Graphics and Applications*, **11** (4) 67 – 74.
7. L. HARDMAN, and A. VAN BOLHUIS (1994). An Interactive Multimedia Management Game. To be published in *Simulation and Gaming*.
8. L. HARDMAN, G. VAN ROSSUM, and D.C.A. BULTERMAN (1994). The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. *Communications of the ACM* **37** (2), 50 – 62.
9. 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, 183–191.
10. L. HARDMAN, G. VAN ROSSUM, and D.C.A. BULTERMAN (1993). Structured multimedia authoring. In Proceedings: *ACM Multimedia '93*, Anaheim CA, 283 – 289.

11. I. Herman, G.J. Reynolds and J. Davy (1994). MADE: A Multimedia Application Development Environment. *CWI Quarterly*, elsewhere in this issue.
12. M.E. Hodges, R.M. Sasnett and M.S. Ackerman (1989). A construction set for multimedia applications. *IEEE Software* **6** (1), 37 – 43.
13. W.E Mackay and G. Davenport (1989). Virtual video editing in interactive multimedia applications. *Communications of the ACM* **32** (7), 802 – 810.
14. R. Macneil (1991). Generating multimedia presentations automatically usingTYRO, the constraint, case-based designer's apprentice. In Proceedings: *IEEE 1991 Visual Language Workshop*, 74 – 79.
15. J.A. Martin (1993). Top 10 CD-ROMs. *MacWorld*, 98 – 105.
16. M. Mills, J. Cohen and Y.Y. Wong (1992). A magnifier tool for video data. In Proceedings: *CHI '92*, Monterey CA, 93 – 98.
17. 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), 67 – 83.
18. T. Nott (1993). Microsoft Encarta. *Personal Computer World*, 380 – 385.
19. 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), INRIA France, 38 – 51.
20. 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, 183 – 188.
21. B. Rubin and G. Davenport (1989). Structured content modeling for cinematicinformation. *SIGCHI Bulletin* **21** (2), 78 – 79.
22. J. Schorr (1993). First-time authoring. *MacWorld*, 106 – 113.
23. J. W. Semich (1992). Multimedia tools for development pros. *Datamation*, 90–95.
24. G. Smarte (1993). Two powerful and friendly multimedia editing programs. *PCWorld*, 114.
25. N. West (1993). Multimedia Masters: A guide to the pros and cons of seven powerful authoring programs. *MacWorld*, 114 – 117.