6 A Hypermedia Authoring Environment: CMIFed

In this chapter we describe the hypermedia authoring environment CMIFed. We go through the requirements stated in the previous chapter and state whether CMIFed satisfies these requirements or not. We discuss the reasons for the latter case.

6.1 Introduction

This chapter describes an existing hypermedia authoring system, built in the spirit of the requirements derived in the previous chapter. For each requirement we state whether or not it has been implemented. If it has not been implemented we state the reasons why.

While the requirements were argued with some rigour in the previous chapter, one issue was not considered, that of the mutual compatibility of the requirements. Many of the requirements are independent, and can thus easily be satisfied within the same system, a small number, however, are conflicting. We highlight where conflicting requirements had to be resolved and what the advantages and disadvantages are of the implemented solution.

CMIFed has a number of separate, but communicating parts—a composition editor called the hierarchy view, a resource-based view called the channel view, a hyperlink editor and a playback environment, referred to as the CMIF player. These are shown with their correspondence to the data, resource, component and document layers in Fig. 6.1. We discuss each of these layers in its own section. The greatest deviations from the previous chapter are that there is no separate link component editor, that there are no separate style or format resources other than those specified in the channels and the atomic components, and that attributes are currently not implemented.

Sections 2 to 5 reflect the structure of the previous chapter: data layer, component layer, document layer and resource layer. The final section draws conclusions from the current editor and looks to future generation hypermedia editors.

6.2 Data Layer

The data layer, shown at the bottom of Fig. 6.1, contains the data resources external to the document itself. CMIFed supports the majority of the data layer requirements, in particular, the separation of data from the atomic component, and the selection of a part of an image item as the content. CMIFed does not, however, provide an overview of a large number of media items. In the most

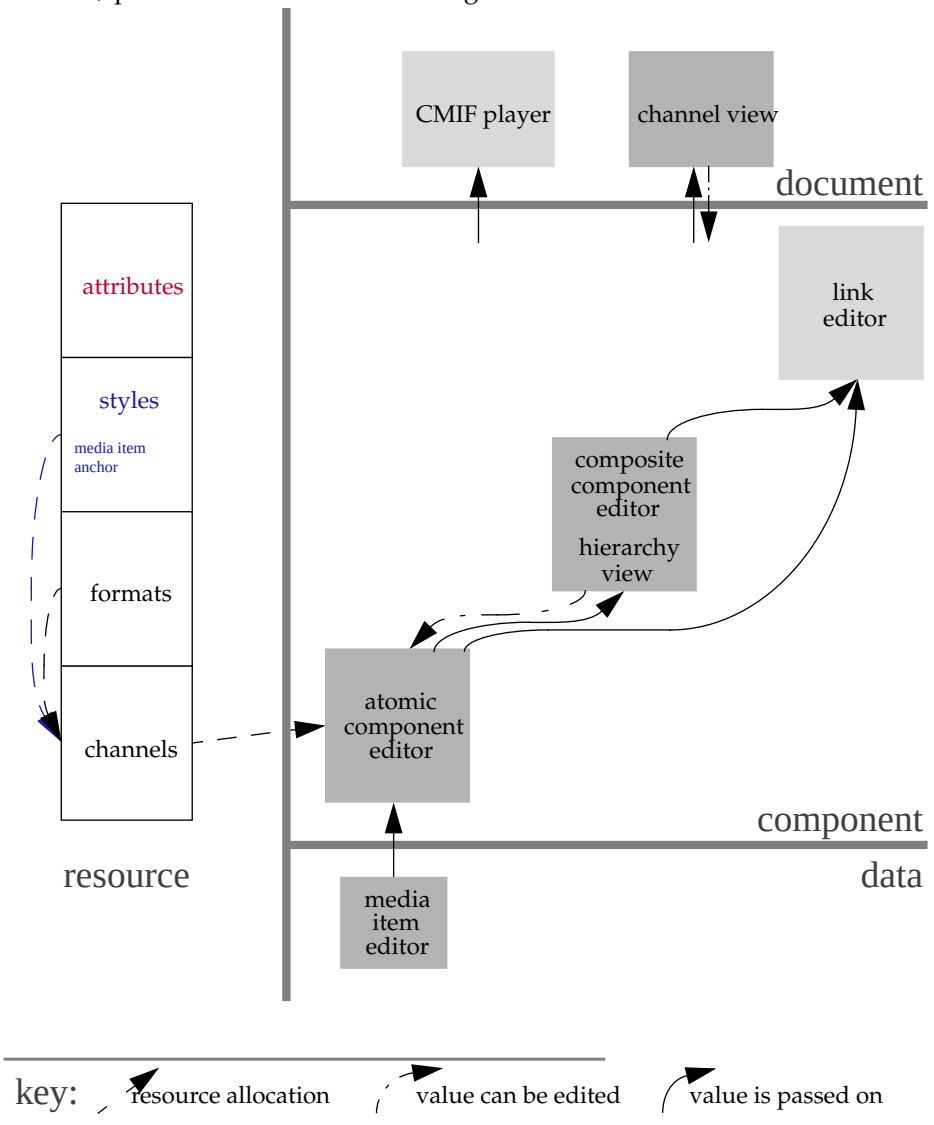


Figure 6.1. Overview of data fl ow for four layers

common windows environments there are other tools that can be used for identifying the content, which means the functionality does not necessarily have to be supported within the editing environment itself.

CMIFed has a separate resource of permissible channel types, but stores no explicit list of permissible data formats corresponding to these. The CMIF player uses the same resource of channel types.

6.2.1 Media items

In CMIFed media items can be any of a number of media types including standard text, image, audio and video, also specialist media types such as a graph. They can also contain a scripting language which is "played" by passing it to an external environment. The store of media items is assumed to be distributed.

Create media item

The only media type CMIFed is able to edit directly is text. All others are created in specialist editors which are not part of the authoring environment. These editors can, however, be accessed directly from within the authoring environment.

The data formats are accessed via the channel types. For example, an image channel can interpret a number of common image formats including TIFF, JPEG, and RGB. These formats are not specifi ed explicitly in the editor code, but in the libraries available to the underlying language.

Select media item

Media items are displayed in CMIFed by supplying a list of fi le names in a dialog box. There is no further support for, for example, displaying icons of images or videos. There is no direct coupling of the fi lename in the dialog box with a viewer to allow the author to select a candidate fi le and then play it to see if it is indeed the one required. This is possible, however, once a media item has been included in the presentation by playing the appropriate part of the presentation.

Edit media item reference

CMIFed supports selection of part of a media item only for images. The reasons for this are media dependent, so we discuss them separately.

- For the text case, the data size for inclusion of a text item in a multimedia presentation is small, in contrast to a text-based application, such as a thesis, where text passages are considerably longer. The amount of effort required to ensure a robust method of specifying part of the text is out of proportion compared with the ease of copying the text item and editing it for each re-use.
- For audio, this could easily have been implemented for the format used in the system (AIFF) since markers are available within the data fi les. For the applications created to date, however, this has not been suffi ciently important to warrant its implementation.
- For the case of video, it would be useful to be able to specify the beginning and end frames of a video sequence, rather than editing the source video.

This did prove to be a problem while authoring, and required tedious editing of the source material to obtain the correct clips for the presentation. A hindrance to implementation are video compression formats which rely on the existence of key frames at regular intervals in the data file, such as MPEG. If the author wishes to start a sequence between key frames the video player has to first play and throw away the unwanted frames before displaying the starting point to the end-user. If the video is cropped (a spatial selection of the video) then it is unlikely that this can be done by the server, so that there is no saving in the amount of data sent to the client. This leads to no real benefit of selecting part of the video item. In conclusion, only temporal selections from video items would be useful, although the data format may not facilitate selection.

6.3 Component Layer

The component layer contains the objects that record all the information needed for describing the behaviour of a presentation, in particular the timing relationships, layout information and navigation routes. The components refer to resources which complement the information stored in the components, in particular the media items, styles applicable to the data and information about the data. The components stored in this layer are atomic, composite and link.

6.3.1 Atomic components

CMIFed supports the creation, deletion, cutting, copying and pasting of atomic components. An atomic component consists of content, anchors, presentation specification and attributes. The data flow in and out of the atomic component editor in CMIFed is shown in Fig. 6.2. The corresponding dialog boxes are shown in Fig. 6.3.

Content

The content of a CMIFed atomic component is a reference to a complete media item, given as a URL in Fig. 6.2(a), or can be a text item included directly. For an image item, part of the image can be designated as the content, shown as the "Image crop" field in Fig.6.2(b). CMIFed does not support alternative content for an atomic component, but at the channel level. We discuss this further in Section 6.4.5.

Anchors

CMIFed allows the specification of an anchor identifier for all media types. We discuss these per media type. In the case of text, the identifier is specified by the author as the name part of the anchor definition within the text content. For the other media, an anchor identifier is specified initially by the system (when the author creates a new anchor) and can be replaced by the author.

An anchor value can be specified within text or associated with an image or video. A text anchor value specifies a single text string. An image anchor value

specifies a rectangular part of the image. A video anchor value specifies the complete video. An audio file can have markers specified within it, but these are not supported by CMIFed as audio anchor values. An enhancement to the current environment would be to support the use of audio markers which could be used as the ends of synchronization arcs or links. CMIFed does not support the specification of text anchors separately from the text data. In contrast, for images and video only the separate specification of anchors is supported.

Text anchor values cannot overlap because of the syntax of the text mark-up used. Overlapping anchors for image anchors can be specified. There is, however, no resolution of the problem of which link to follow where two or more anchor values overlap—the interaction is ignored. To resolve this issue a method

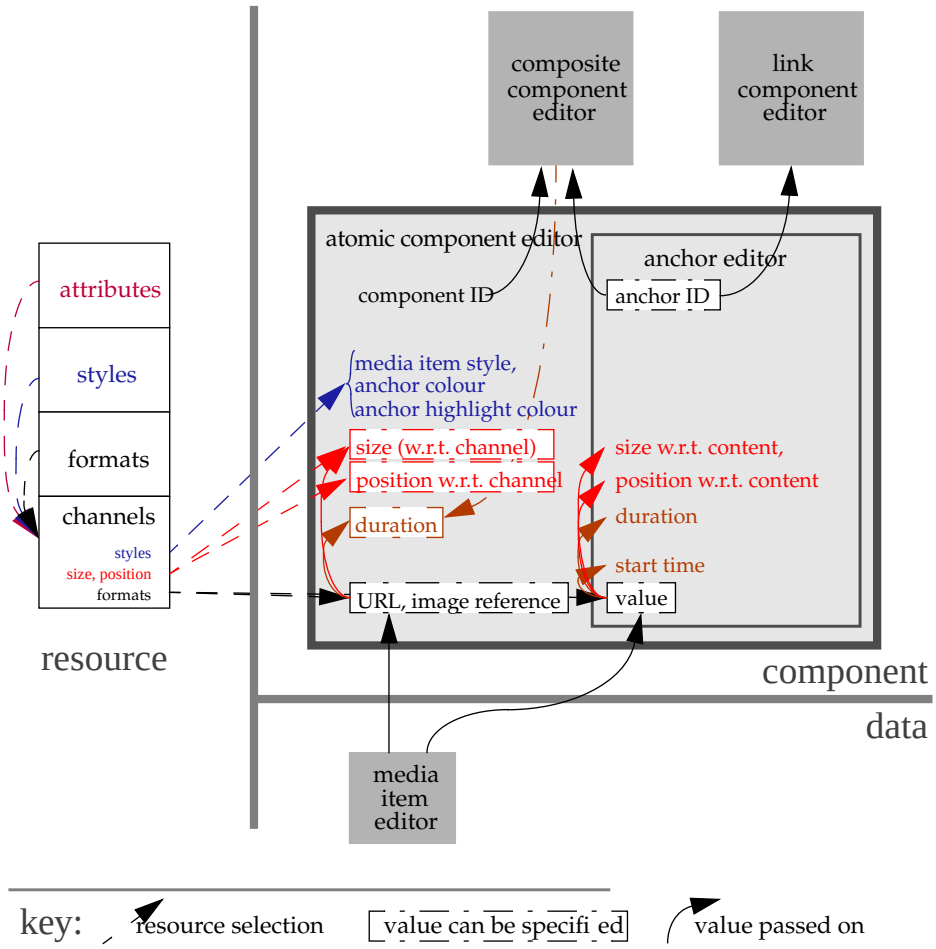


Figure 6.2. Data flow in and out of atomic component editor

for choosing among the links that can potentially be followed, such as an intermediate menu [GaSM86], would need to be developed.

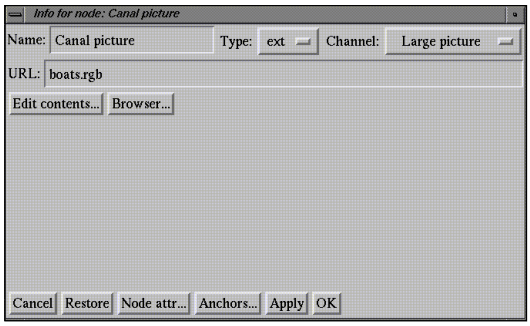
Attributes and anchor-specific presentation information are not supported within CMIFed. The consequence of the latter is that all anchors within the same media item have the same presentation style. This affords consistency, but does not allow, for example, the distinctive highlighting of individual anchors.

An anchor in CMIFed can be of a number of different types: normal, auto-firing, destination only, pause.

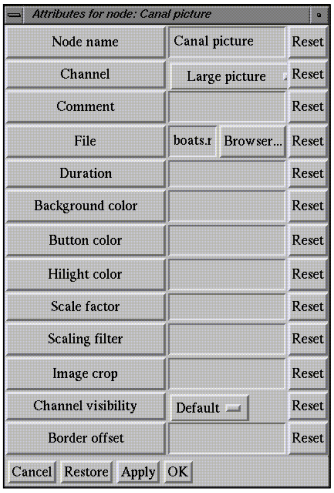
- A *normal anchor* can be used as the source and/or destination of a link, and is specified as described above.
- An *auto-firing anchor* is one that triggers the following of the associated link automatically when it is displayed.
- A *destination only anchor* can be used only as the destination of a link.
- A *pause anchor* stops the progression of the presentation until the user clicks on the anchor. Pause anchors can be used to simulate pre-arming on following a link. If the destination of a link could always be displayed instantaneously then they would not be needed.

Timing

Text and image atomic components can be assigned an absolute duration. The duration for audio and video atomic components is the intrinsic duration of the media item. Scale factors in terms of the intrinsic duration of audio or video are not supported, and neither can an absolute duration be specified. A simple enhancement to the current environment would be to implement the following



(a) Node information



(a) Node attributes

Figure 6.3. CMIFed atomic component information

options for specifying absolute or relative durations. If the specified duration is shorter than the intrinsic duration then the media item can be stopped before it reaches its end-time, but not be played faster. If the specified duration is longer then the media item can remain visible or be repeated, but not be stretched.

An atomic component of any media type can be assigned an indefinite duration, called a *pause node*. The component will continue to play indefinitely in the player, and can only be made inactive by following a link to another part of the presentation. This is useful for keeping a presentation on the screen when all its parts have finished playing.

The duration of a non-continuous medium atomic component in CMIFed does not necessarily have to be specified on an individual basis but can be calculated from the surrounding hierarchical structure. We discuss this further in section 6.3.2.

Spatial layout
CMIFed bases it layout on channels, where each atomic component is assigned a channel specifying the outer extents of the size and position of the component. Size for images and video can be specified as being the intrinsic size of the content, relative to the size of the content, or as filling the channel (preserving aspect ratio). The position of the content can be specified with respect to the channel, including outside the bounds of the channel, although not outside the bounds of the containing window. Text fills the specified width, but is not scaled to the height of the channel.

In the CMIF player, the size specified in an atomic component overrides that specified in the channel. Position of atomic components is dependent on the associated channels, so that changes in the relative positions of atomic components need to be resolved at the channel level.

Styles
CMIFed uses channels as a style resource and supports media item and anchor styles but not transition effects. Media item styles include font for text and background colour for all visual media types. Anchor styles specify the colours for displaying an anchor and for highlighting the selected anchor.

In the CMIF player, the style specified in an atomic component overrides that specified by the channel.

Attributes
Attributes are not explicitly supported from within CMIFed.

Discussion
CMIFed has a number of options for specifying the sizes and positions of atomic components. One option the system supports is the scaling of images relative to the intrinsic data size. This feature is seldom used, which leads us to believe that it is not very useful. This is probably because the more important size is the screen display rather than the image itself. A potentially more useful scaling

algorithm is to scale to fit the channel, but for large images and video to restrict the potential sizes to multiples of the intrinsic size, thus speeding up the scaling process. A feature which CMIFed does not support, but which would enhance the scalability of presentations considerably, is the scaling of font size. This would require an author to specify the font size in relation to the size of the channel, using for example height or some combination of height and width, rather than as an absolute point size. The font size could then be calculated appropriately. The scaling algorithm should deal with changes in the size of the channel as well as changes in its aspect ratio.

Transition effects are not supported in CMIFed. This is one of the major deficiencies of the current editor since most commercial multimedia authoring systems support transitions.

Attributes on atomic components and anchors are not supported within CMIFed. At the time of writing there is no attribute resource associated with CMIFed from which to choose attributes. In order to support this feature fully a complete environment needs to be created rather than just an addition to the CMIF document format. Work is being carried out in this direction, [BuRL91].

In summary, CMIFed provides comprehensive support for atomic components in all areas except transitions and attributes. We consider one of CMIFed's important contributions to be the introduction of atomic components to a multimedia authoring environment. By separating out data dependencies from the rest of the environment they allow an author to edit the presentation in a uniform manner and, by allowing the association of different properties with the media items, multiple uses can be made of a media item.

6.3.2 Composite components

CMIFed supports the creation, deletion, cutting, copying and pasting of composite components. A composite component consists of a list of children, anchors, a presentation specification, and attributes.

Children

CMIFed supports two forms of temporal composition—parallel and sequential—and one form of atemporal composition—choice. Parallel children are started together and sequential children are played one after another. The data flow in and out of the editor for parallel and sequential components is shown in Fig. 6.4. In choice composition only one child can be played at any one time. The data flow in and out of the choice composite component editor is shown in Fig. 6.5.

CMIFed supports the top-down and bottom-up creation of composition structure. Composite components can be copied for inclusion in other parts of a presentation, but cannot be re-used. The hierarchy view, used for visualizing and navigating the composition structure, is illustrated in Fig. 6.6. While the hierarchy view is not strictly a time-based view, it shows parallel children next to each

other and sequential children one above the other. CMIFed also provides dialogs to view the properties of each composite component, Fig. 6.7.

Activation state

The initial activation state of each child of a choice component is either inactive, or a single child can be nominated. This is termed the bag index in the editor and is shown in Fig. 6.7(b).

Anchors

CMIFed does not support composite anchors, although it does support an anchor belonging to a composite component which serves as the destination of a link, called a destination anchor. Neither does CMIFed support the assignment of anchor properties in a composite component. The editor does, however, partially support equivalent behaviour by different means. A composite source anchor

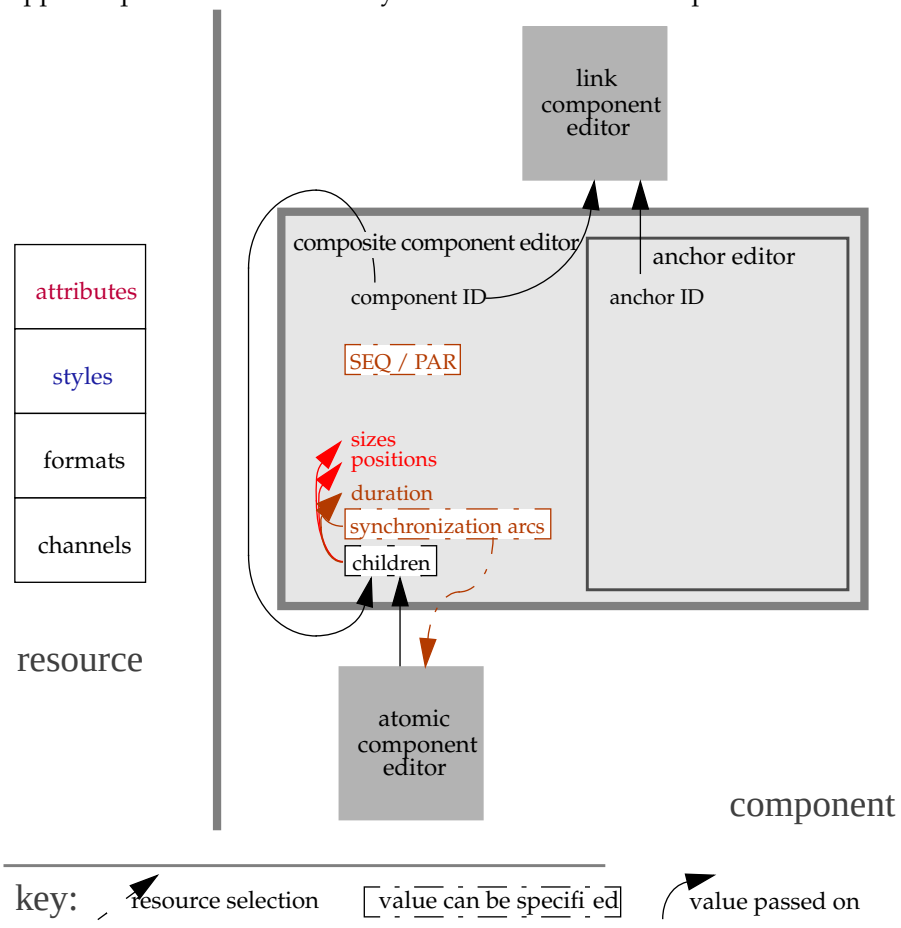


Figure 6.4. Data fl ow in and out of temporal composite component editor

can be simulated by creating links from each of the individual atomic anchors to the same destination. A composite destination anchor can be approximated by linking to a composite component containing all the destination components, since CMIFed links have only one destination component. In either case, behaviour involving all parts of the composite anchor, such as highlighting the anchors on following a link, cannot be simulated. CMIFed is not able to benefi t from the richer semantics of composite anchors, but, since CMIFed currently provides no support for attributes, this is irrelevant.

Timing

In CMIFed, temporal relations are captured in parallel and sequential composition along with any synchronization arcs. The duration of a parallel or sequen-

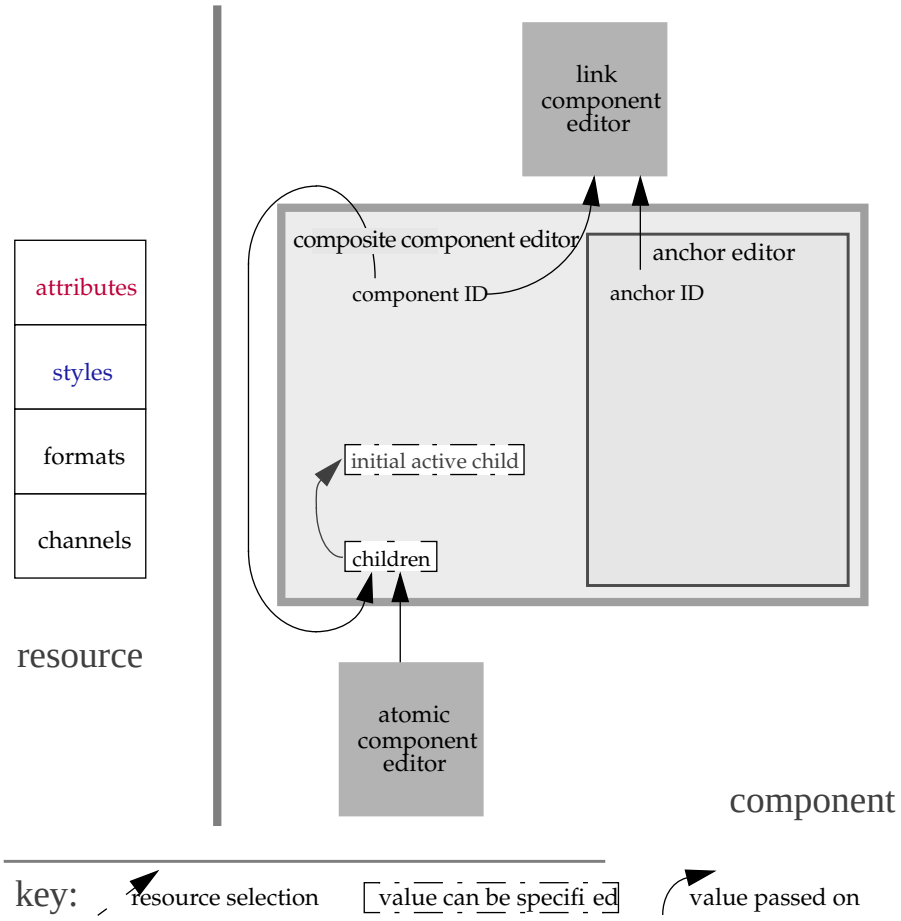
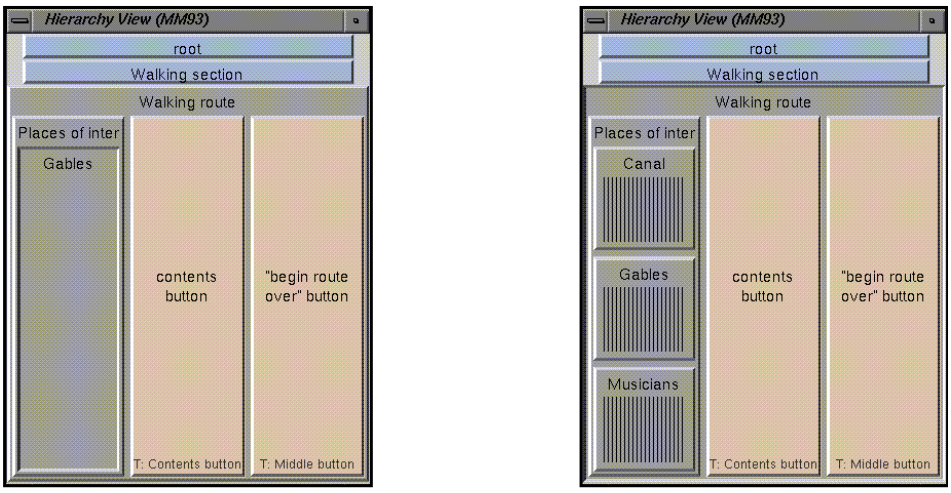


Figure 6.5. Data fl ow in and out of atemporal composite component editor



(a) (b)

The hierarchy view displays composite and atomic components—the latter being leaf nodes of the hierarchy. The number of titles bars shows the level in the hierarchy. Time is not represented uniformly, but there is an ordering: objects side by side are played in parallel and objects one under the other are played sequentially. "*Contents button*" and "*begin route over button*" are atomic (text) components.
(a) "*Places of interest*" has one child component "*Gables*" which is an empty composite component.
(b) "*Places of interest*" has three children, each a composite component containing hidden structure.

Figure 6.6. CMIFed hierarchy view

Figure 6.7. CMIFed composite component information and attribute dialogs

tial composite component is calculated from the durations of its children and the synchronization constraints that exist between their descendants.

CMIFed does not support the specifi cation of a duration for a temporal composite other than that derived from its descendants. This means that an author is not able to alter the duration of a composite as a whole. A low cost improvement to the current functionality, but without making a change to the document format and the player behaviour, is to implement a facility in the editor which goes through the timing in all the descendants of the composite and scales the durations by the appropriate factor. This would also require scaling the duration of atomic components for audio and video (e.g. using a low implementation cost cut-off and repeat).

CMIFed supports the specifi cation of synchronization constraints between atomic components belonging to the same temporal composite, Fig. 6.8, but not between composite components.

CMIFed supports the derivation of duration from the surrounding composition hierarchy and synchronization arcs, although synchronization arcs cannot be defi ned with respect to a composite.

Spatial layout

CMIFed assigns position and size only via atomic components in relation to channels.

Styles and Attributes

CMIFed does not support the assignment of styles or attributes to a composite component. The reasons for the latter are the same as those for an atomic component.

Discussion

Our experience is that while parallel and sequential composition are specializations of temporal composition, that they are such a natural refl ection the narrative structure of a presentation that they should be supported explicitly. Authors

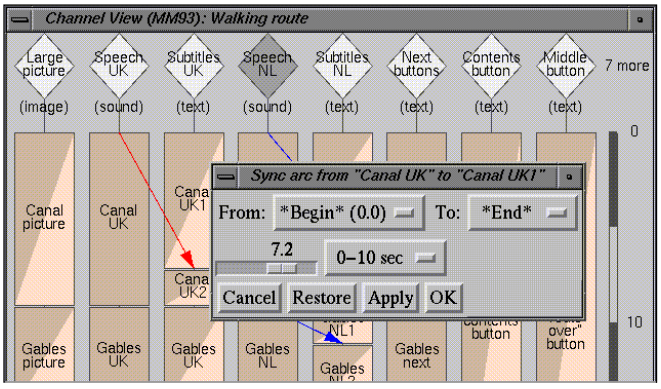


Figure 6.8. Creating a synchronization arc

tend to use synchronization arcs only occasionally for refining the timing beyond that specified using sequential and parallel composition. We strongly recommend explicit authoring support for parallel and sequential composition.

A disadvantage of supporting both parallel and sequential composition is that the author is required to specify structure even when this seems to be unnecessary, for example where parallel and sequential nodes are alternately nested to a number of levels deep. It is unclear whether a different authoring interface could help, or whether this stems from the intrinsic complexity of the presentation's structure and cannot be avoided.

CMIFed permits the copying of composite components for inclusion in other parts of a presentation, but not literal re-use. If re-use were implemented, however, the power of the feature would be lost since CMIFed does not support the assignment of styles, spatial or timing information to composites, so that re-use would always be literally the same and variations in, e.g., style could not be made. To include a re-use feature would require the resolution of a number of functionality issues as well as the design of an appropriate user interface. The functionality issues requiring to be solved for any system supporting re-use, not only for CMIFed, include the following:

- Should there be one master version of the composite which can be referred to, or are all copies of the composite equivalent, e.g. similar to UNIX file linking.
- How can the shared composite component be assigned different styles per instance? Does another composite component need to be created around the re-used composite to which different styles can be assigned?
- What is the priority of the styles defined in the different places of the document, in CMIFed the descendant atomic components, the channels and the instances of the composite?
- How can the author find out in which places in the structure the composite is being re-used?

CMIFed supports the specification of synchronization constraints between atomic components belonging to the same temporal composite, but not between composite components. This results in a set of temporal constraints less rich than those described in, e.g. [BuZe93]. The reason for this omission in the current environment is that there is no user interface which shows structure and time in the same visualization. In CMIFed the composition structure is visualized using the hierarchy view and timing is visualized in the channel view. Because channel assignment is entirely independent of structure, there is no way of showing time, structure and channel assignment in a single (two-dimensional) view. This point of view contrasts with a system which does show time and structure in the one view [Acke94], but does not use channels. This visualization, the essence of which is reproduced in Fig. 6.9, is, however, ambiguous since it is does not show whether a second child (C) is a sibling of the first child (B) or a grandchild of the

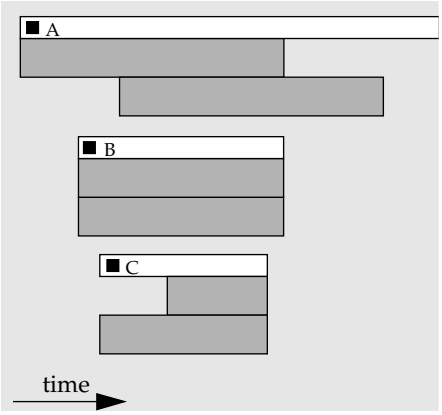
parent (A). In order to show structure and time unambiguously in one view three dimensions are needed.

An important advantage of the CMIFed environment is that the size and duration of an atomic component do not have to be specifi ed laboriously per component, but instead duration can be derived from the surrounding structure and layout inherited from the associated channel.

In summary, visualizing, editing and navigating the composition structure is supported in the hierarchy view. Timing information is derived from the composition structure. If low cost scaling of the duration of audio and video were implemented for atomic components then scaling the duration of a composite would be a further low cost enhancement. In order to provide an authoring interface to creating synchronization arcs to composites then a user interface including time and structure needs to be designed.

6.3.3 Link components

Authoring links within CMIFed is carried out in the link editor, which serves both as link manager and link component editor. The emphasis is perhaps more on link management, since there are few options for editing a particular link—the interface supports the selection of anchors as the predominant means of interaction. CMIFed supports the creation and deletion of individual link components. A link component consists of specifi ers, a presentation specifi cation and attributes. The data fl ow in and out of the link component editor is shown in Fig. 6.10. CMIFed does not support the high-level creation of links, although initial design work has been carried out in this direction [BuRL91].



The composition structure is shown as labelled white horizontal bars. The atomic components are grey boxes. B is a child of A. It is ambiguous whether C is a child or a sibling of B.

Figure 6.9. Structural view with timing is ambiguous

Specifiers
In CMIFed a link can have only two specifiers, which prevents the creation of a link with multiple destination components. A similar effect can, however, be achieved through structuring the destination component. For example, the destination component may have children displayed in multiple windows, since spatial layout and structure are independent of one another. The destination component may contain several choice components, each of which has the potential of starting up a presentation independent of the others.

A specifier is created by selecting two existing anchors within the same or separate documents then creating a link between the two anchors. For each specifier the following are required:

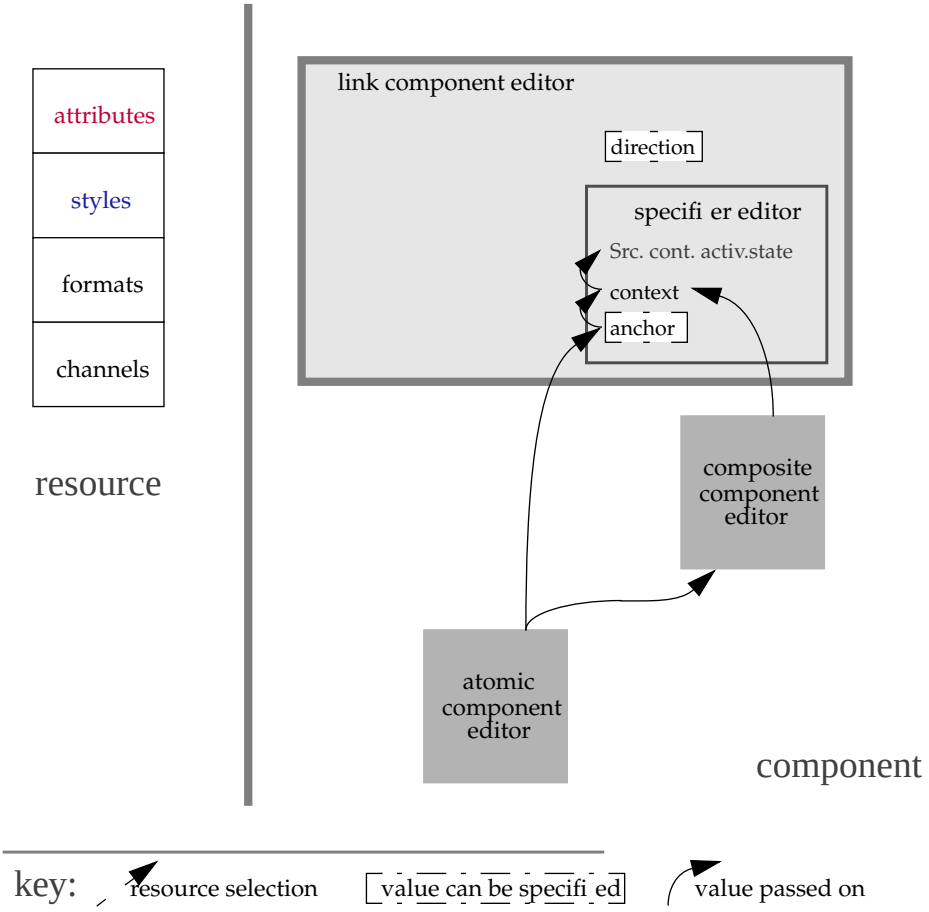
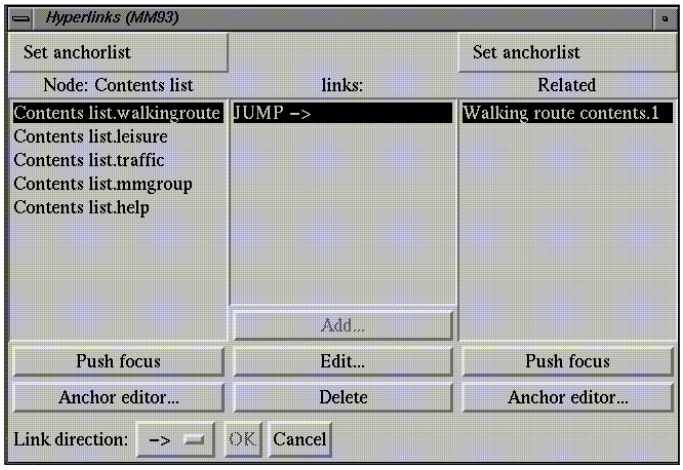


Figure 6.10. Data flow in and out of link component editor

- *Anchor reference*
  The author is able to use either the hierarchy or the channel view to select a component, and then pass this component to the link editor. The list of anchors belonging to the component is displayed. The author can select one of these to act as an end of the link. The author can carry out a similar process to select another anchor to act as the other end of the link. CMIFed also ensures the same anchor cannot be specified as both the source and destination of the link.
- *Direction*
  The author selects whether the link (between the two already selected anchors) is forwards, backwards or bidirectional. The editor checks whether the direction is valid, i.e. disallowing links from a destination-only anchor.
- *Context*
  The source context is derived from the composition structure surrounding the source anchor—the source context is the child component of the ancestor choice component lowest in the hierarchy that contains the anchor[1]. This forces the author to create the structure so that following the link gives



The highlighted text on the left shows the name of the source component and the source anchor. Similarly, the destination component and anchor are shown on the right. The middle column shows the direction of the link.

Figure 6.11. CMIFed link editor

1. The source context is a temporal composite that is an immediate child of an atemporal composite component.

the correct behaviour. An improvement to the current functionality would be to allow the author to specify any ancestor choice component as the source context rather than being restricted to the lowest in the hierarchy. An improvement to the current interface would be to aid the author in creating the correct document structure for the desired link behaviour.

- Activation state
  The source context activation state is dependent on the relation between the source and destination contexts. The source remains playing if the destination context is in a separate choice component, and is replaced otherwise. Specifying that the source context should pause is not possible because a means is needed for reactivating it. All active presentations are, however, controlled by a single player control. Waiting for the destination context to finish playing is undesirable, because the player needs to be aware of when the appropriate destination has finished playing.
  The destination context play/pause state is not specifiable. This also requires a control for each active presentation. Note that, since windows are independent of a single presentation, if there are multiple controls it has to be made clear to the end-user which player control is coupled to which active presentation.
- *Style*
  CMIFed does not support styles for specifiers.

Timing
CMIFed does not support transitions on links, and in particular does not allow the specification of a duration.

Spatial layout
The layout of the destination context is determined by the channels associated with the atomic components in the destination context.

Styles
CMIFed does not support the specification of transition effects.

Attributes
CMIFed does not support the assignment of attributes to a link component.

Discussion
While the current environment provides some support for links, it becomes cumbersome for an author to specify the document structure for defining ends of a link. Even when this has been done and the author has included multiple independent presentations in the destination, the author is unable to specify a destination anchor within each of the presentations. A low cost improvement to the current link editor would be to support the specification of a destination anchor per choice component in the destination.

When selecting the anchors to act as link ends it is often useful to be able to see the source and destination components in the hierarchy view. This, however,

requires the presence of two hierarchy views each with a different focus, or two different foci within the one view. The whole environment is currently built around the notion that there is one focus, and that this can be passed among views (hierarchy, channel, player and link editor). The link editor compensates for the lack of multiple foci to some extent by allowing the direct selection of a component from the channel or hierarchy view focus.

Bidirectional links can be specifi ed inCMIFed, but it is our experience that these are rarely used. For a link to be truly bidirectional the source marker and the destination marker need to have a symmetrical relationship with one another. This is not normally the case. In a multimedia environment, links tend to start from a particular point (the anchor) and lead to a new presentation (a number of components). One example of a symmetrical link is in a virtual reality where a doorway connects two rooms. You would expect to use the same doorway to go back and forth between the rooms. Since the doorway connects only two rooms the symmetric link is between both instances of the same doorway—the source and destination anchors are the same doorway, the source context for the link is the room you are in, the destination context is the other room, and the link is bidirectional. If the doorway connected three rooms (!) then there would be a link with all three instances of the door as source/destination anchors and the rooms as the contexts. When entering the doorway from any of the rooms you end up in both destination rooms simultaneously. We do not regard support for bidirectional links as compulsory in a hypermedia environment. Note that following unidirectional links backwards is a different issue and should be supported.

In summary, the link support in CMIFed has not been a major focus of the work. Our fi rst priority for impɒving the link support is on transitions, and then to improve the editing environment for link components. For some additional features we need only a slightly extended link structure using properties the editor already knows about (such as anchor style) and that the player can deal with. There are currently no editors which support the selection of source and destination contexts or transition information explicitly. We remain of the opinion that such editors need to be implemented.

6.4 Document Layer

The document layer allows the author to concentrate on single aspects of the presentation, in particular the presentation's timing, spatial layout, link management and runtime behaviour. Specialist editors should be provided for each of these aspects.

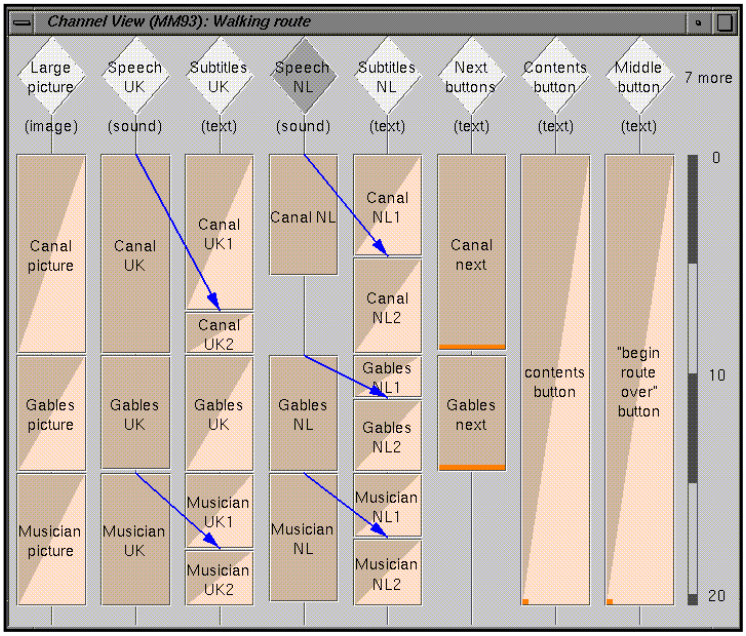
6.4.1 Timing

At the document level, timing is a crucial aspect in authoring a hypermedia presentation. The timing of a presentation is based on the duration and start

time of an event, and synchronization relationships among events. As we discussed in chapter 4, while a timeline gives a useful overview of the timing of the presentation, it is not necessarily the best overall authoring paradigm. We advocate a structured approach to authoring with an associated time-based view. CMIFed derives the timing information from the composition structure and synchronization constraints and displays the result in a time-based view—the channel view.

Duration of event

The event derived from an atomic component has a duration. In CMIFed this can be an intrinsic duration determined from the content, an absolute duration specifi ed by the author, or a derived duration determined by the surrounding composition structure in conjunction with the synchronization arcs. Fig. 6.12 illustrates the channel view showing the durations of events, which components have intrinsic or author-specifi ed durations, and which have derived durations.



Time goes from top to bottom, as indicated on the time bar at the right.
Events with intrinsic or author-specifi ed duration are shown as solidly shaded boxes. Events with duration derived from the composition structure and synchronization arcs are shown with diagonal shading.
The arrows indicate synchronization arcs between events.

Figure 6.12. CMIFed timeline in Channel view

Start time of event
The event derived from an atomic component has of itself no explicit start time. This has to be calculated from the relationships among events in the presentation. In CMIFed this is captured via the composition structure and the synchronization arcs for parallel and sequential composites. For choice composite component the children are not related in any temporal way, so that the start time of any child can be determined only at runtime. This is indicated by using separate channel views for each child of a choice component. The start time of the first child of a choice component is taken to be zero.

CMIFed supports the editing and visualization of synchronization arcs in a timeline view, the channel view, Fig. 6.12. Constraints between structures cannot be specified, as discussed in the previous section on composite components.

When an author defines a synchronization arc that introduces cyclic dependencies then the system warns the author of the presence of a cycle. If the arc is not removed it is ignored both in the channel view and by the player.

Derive timing from structure
In CMIFed timing is derived from the intrinsic durations of events, the composition structure and synchronization arcs. The duration of a sequential or parallel composite is derived from the duration of its children, and the duration of atomic components with no intrinsic or specified duration is derived from the duration of the composite. The latter works best with parallel composition, when the atomic component's duration equals that of the parent composite. This no longer gives reasonable behaviour when a parallel composite contains only children with non-specified durations, such as text and image items, since then the durations of the composite and all its children are zero. If the author assigns a duration to any one of the children then the problem is solved. A sequential composite requires a duration for each of its children.

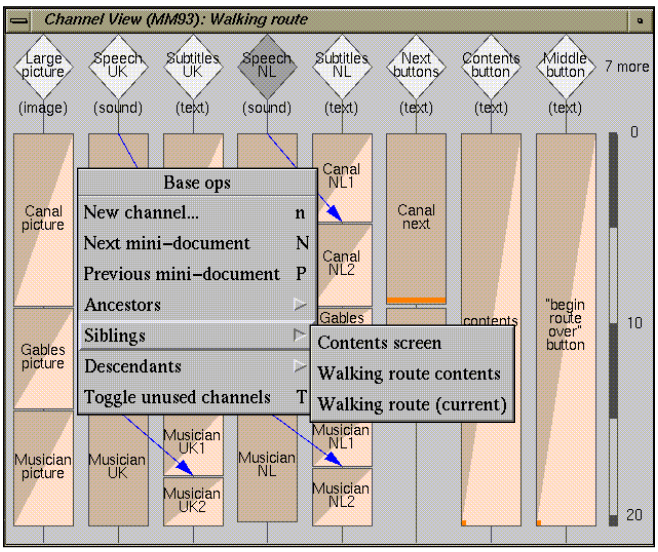
Duration of a link transition
CMIFed does not support the specification of a link transition duration, as discussed in the previous section.

Tempo
CMIFed supports neither the editing of tempo nor the realtime playback of different tempos. The CMIF document model does not include tempo, and it is unclear where the tempo could be stored if the model were revised. It could be stored with a composite component, but this would be equivalent to changing the duration of the whole component. More interesting would be to record *ritardando* and *accelerando* directions. This would require a new structure in the CMIFed composite component to record the rate of time flow for the duration of the composite, plus serious adjustments to the player software, in particular that different active composites could have different tempos.

Applying temporal transformations throughout hierarchy
CMIFed does not propagate temporal transformations throughout the document hierarchy. This may seem to be a defi ciency in the system, but since the timing aspects are calculated bottom-up, the duration of the composite as such is less relevant. If CMIFed supported synchronization arcs to or from a composite, then the duration of the composite could be based on the duration of another component, so that scaling the composite's duration would be a required editing action.

Synchronization specification
CMIFed supports only individual specifi cation of synchronization arcs. In order to provide support for higher-level specifi cation, an interface for each particular case would need to be implemented. Our experience with the applications up until now is that these have not been suffi ciently important, but for a language training application, for instance, such a tool would be essential.

Navigate timeline
The timeline for a single presentation may become long and unmanageable so that there needs to be some way of changing the scale of the timeline. CMIFed



The left-hand menu shows a list of commands currently available. Among these are the navigation commands *Next mini-document*, *Previous mini-document*, *Ancestors*, *Siblings* and *Descendants*. *Siblings* has been selected and the three sibling names are displayed, including the current timeline.

Figure 6.13. Navigating multiple timelines

does not provide facilities to zoom in on the timeline view, but allows instead navigation through the multiple timelines, Fig. 6.13.

Discussion

CMIFed supports the visualization and editing of synchronization constraints via the channel view. Much of the temporal information displayed in this view is derived from the composition structure created in the hierarchy view.

In the channel view there is no visual distinction made between events with intrinsic or author-specified duration, Fig.6.12. This has not been a problem in our experience, possibly because intrinsic duration in the current environment applies only to audio and video items and specified duration only to text and image items. The solid boxes give the impression that the duration is fixed. The diagonally shaded boxes give the impression that the duration can stretch. From our experience with using the environment this is sufficient information for the author.

We consider one of CMIFed's important contributions to be the functionality provided for deriving durations within a presentation. An author may specify the media items, and only when there is insufficient information to deduce the durations of events is the author required to specify a duration explicitly. Our experience is that the duration derivation makes such natural default decisions that this feature is virtually transparent to the author.

While the channel view misses a zoom-in facility, our experience is that the presentations are not very long. While long multimedia presentations can be created, most often there is some form of interaction expected from the user before introducing new information. If there is one long movie then it is also likely that there are few dependencies, so that a small channel view would be sufficient. Up until now we have experienced no problems with the temporal navigation facilities, but we may find that as presentations become larger that a zoom facility on the channel view timeline becomes necessary.

In summary, the channel view provides a separate timeline based view for each child of a choice component and supplies facilities for navigating among the children. Within each timeline an author is able to define synchronization constraints between atomic components. One of the powerful authoring aspects of the CMIFed environment is the derivation of durations of components from the composition structure and intrinsic durations of continuous media items.

6.4.2 Spatial layout

At the document level, spatial layout is a crucial aspect in authoring a hypermedia presentation. The layout of a presentation is based on the size and position of events, and spatial relationships among events. The layout of a presentation in CMIFed is defined by the spatial extent of the events and how these are placed with respect to a channel.

Size of event
The event associated with an atomic component has a size. This can be the intrinsic size of the content, can be specified as a relative or an absolute size by the author or can be derived from an associated channel. CMIFed does not support the definition of size with respect to other events. Visualizing the size of a single event is supported by playing the event. The size of an object is not able to change with time.

Position of event
The event derived from an atomic component has of itself no explicit position. Each atomic component has an associated channel which specifies an area in relation to a layout channel or window. Position information specifies where the event is placed in relation to the channel (with caveats for the different media types as stated previously). The author is supported by being able to edit the size and position of a channel extent while being able to see all the other channel extents.

Visualizing the position of one event is supported by playing the event. A visual overview of the layout of all events playing at any one time is provided via the player. The channel view shows which events are played when on which channel, but does not give a direct overview of the positions of the events on the screen.

CMIFed supports neither the creation nor the visualization of paths. Given that position with respect to a channel is already implemented there is no fundamental reason for not implementing paths with respect to a channel.

Position of transition
Position is specified via channels, so that the spatial placement of the link contexts is already specified.

Channels
Rather than having to specify the position for every event in the presentation, some method of specifying layout information at a higher level saves the author work. CMIFed supports the use of channels which pre-define areas in a window. Sizes and positions of channels can be viewed in the player. Creating a consistent layout is also easier because events assigned to a channel are displayed at the same, approximate, position. Authors are able to specify how the size and position of the event relates to the channel as follows:

- the placement of an event in relation to a channel is possible only for image items, as discussed previously in Section 6.3.1;
- an event can be scaled to the size of the channel or can retain the intrinsic extent determined by the content;
- only scaling that preserves the aspect ratio is supported.

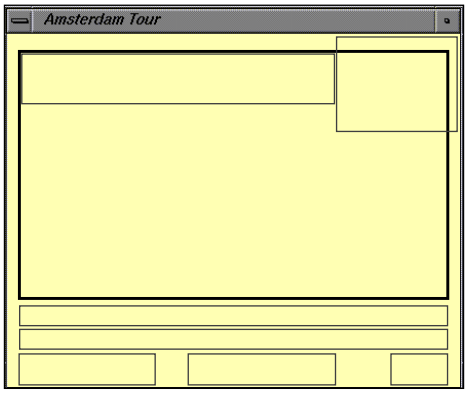
The extents and position of a channel are defi ned with respect to a layout channel. An author is able to edit a channel's size and position while viewing other channel extents in the same layout, Fig. 6.14.

Derive spatial layout from structure
CMIFed does not derive spatial layout from structure, i.e. it does not support the automatic assignment of channels to atomic components in the document hierarchy. This is entirely reasonable since the hierarchical structure is a temporal structure and is thus independent of the layout structure of the presentation. There is of course some relationship, since all items played at the same time need to appear somewhere on the screen, but the relationships cannot be deduced from the temporal information alone.

Applying spatial transformations throughout hierarchy
CMIFed applies spatial transformations throughout the layout hierarchy. If the size of a layout channel is changed then the size of all the other related channels changes in proportion. This action can also be carried out at runtime.

Navigate layout
The layout of the presentation cannot be viewed at different points on the timeline at once. CMIFed does thus not support the direct comparison of two or more different layouts. This has not been a problem to date, possibly because reusing channels guarantees consistent layout where needed, and completely different layouts do not generally have to be compared with one another.

Discussion
The layout of a presentation can be visualized by running the presentation. This method, however, is not a good way of getting an overview of the layout



The black outline shows the channel being edited. The grey outlines show the sizes and positions of sibling channels belonging to the same layout.

Figure 6.14. CMIFed channel layout

throughout the presentation. Visualizing the position of the objects with respect to time requires a visualization in three dimensions, although the three dimensions can be projected back to two [OgHK90]. A view such as that in Fig. 6.15 may be of use to the author, but would require a considerable amount of implementation effort.

If the size of a layout channel is changed then the size of all its dependent channels changes in proportion. We consider one of CMIFed's important contributions to be that this action can be carried out at runtime. There are no other systems that we know of where not only the size of the presentation can be changed at runtime, but also the aspect ratio of the presentation can be changed. The way CMIFed handles channels means that, for not too drastic changes in aspect ratio, the presentation remains visually acceptable. This is similar to textual web browsers where the size and aspect ratio of the window can be changed. With the stricter spatial layout specifi cations for multimedia the problem becomes more diffi cult.

In summary, layout support is provided using channels which facilitate consistent specifi cation of layout and enable support for resizing the complete presentation at runtime.

6.4.3 Link management

A hypermedia author is concerned with the creation of a presentation narrative, which, while supported by composition of components, also requires the specifi cation and maintenance of links among these components. The author needs to check that the possible paths through the presentation are meaningful to the end-user, and whether each path is complete or still under construction. For each link the author needs to verify the source and destination contexts and the transition. When the number of presentations and links becomes large then not only the end-user but also the author can become lost in hyperspace. We recommend the provision of a link management system to aid the author in organising the

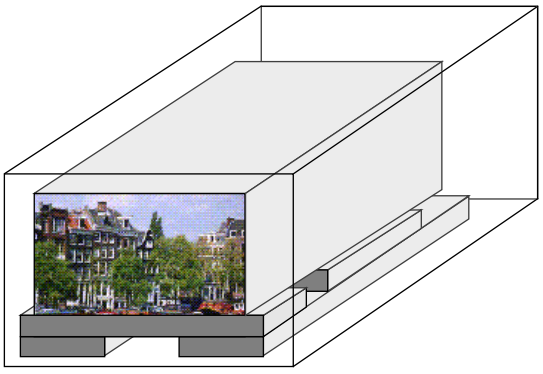


Figure 6.15. 3D layout view

creation and maintenance of links. CMIFed provides limited link management facilities via the link editor, shown in Fig. 6.11.

Find incomplete links
Links can be created only between two anchors, so that incomplete links cannot be created. If an anchor which is being used as the source or destination of a link is deleted, the system also deletes the incomplete link.

Check complete links
An author is able to check the links in a presentation by following each of them individually or by using the "related anchors" facility in the link editor. The latter allows the selection of a component in either the hierarchy or channel view and, from within the link editor, displays the list of anchors belonging to the component. An anchor can be selected and all its related anchors and links displayed in the window. For example, in Fig. 6.11, the anchor "Contents list.walkingroute" has one related anchor "Walking route contents.1". Any of the related anchors can be selected and the author can then request to view its associated component in the hierarchy and channel views.

Find unlinked components
CMIFed allows the definition of structures which can never be accessed from the starting point of the presentation, but has no tools for finding out which these are.

Discussion
CMIFed provides limited link management facilities, such as providing a facility for finding anchors linked to one another. For more robust handling of links a complete link management system would need to be implemented.

6.4.4 Presentation Control

The CMIF player is based on a model where separate, but communicating, processes display atomic components. This gives the end-user control over the flow of the presentation as a whole and also over the individual channels. By turning channels on and off an end-user can select, for example, a preferred spoken and written language.

CMIFed allows the author to play any part of the document in the player, specifically an atomic component or a composite component. It is also possible to select a component and "play from here", in which case the player schedules the complete presentation then fast forwards, without displaying anything, until it gets to the selected component and starts displaying from there. Since CMIFed has no explicit timeline the author is not able to specify a time from which to start playing. CMIFed is not able to fast forward while the items are displayed nor play the presentation in reverse. Fast forwarding has been implemented in previous versions of the player and, while it would be a useful feature, our authors have not missed the feature greatly. The current scheduler works only in one direction when traversing the graph of timing relationships of the events in

the presentation. In order to play the presentation in reverse, the approach to creating the timing graph would have to be redesigned. Fast-forward and playing in reverse are features most useful for end-users of the presentation rather than authors. To date, CMIFed has been used by a number of authors, but there have been few end-users, so that we are unable to judge the utility of these features.

While the presentation is playing, CMIFed allows the author to select any media item in it and inquire where this is included in the presentation's structure. This is carried out by allowing the author to point at any currently playing and visible component and push the focus to the hierarchy and channel views. The author can then invoke any of the editing commands as normal.

While the presentation is playing, the author is able to see which parts of the editable representation are currently playing. Five different states are displayed in the channel view: not playing, preloading data, finished preloading data, actively playing, and finished playing but still visible. These are shown using different highlight colours in the channel view, Fig. 6.12.

Discussion

CMIFed supports the inclusion of independent presentations within a document (through the use of choice composition). These each have their own independent player and thus their own independent timeline.

It is not possible to see which of the separate presentations are currently active. This means that while the author can select a presentation to see which parts of it are active, they cannot see which of the collection of presentations is active. The reason for this is that only one presentation can be viewed at a time in the channel view, and that there can only be one instantiation of the channel view, because of the push focus mechanism. A possibly useful improvement would be to show in the hierarchy view which of the components, or their descendants, are active. While this may be useful for the author, the runtime implications are significant, since for every zoom in or out in the hierarchy view the system has to compute which shading to use for each visible component. Having the channel view open while running a presentation already puts a burden on the processor resources.

There is a restriction that when the author edits a playing presentation that the player has to be stopped. While this is annoying for the author, if the scheduler were not to stop and recalculate a new schedule then there could be inconsistencies in, for example, parts of the presentation which had already played. This problem is less obvious in environments where the author is forced to stop the presentation before returning to the authoring environment. In CMIFed the author is at least able to view all the characteristics of the components while the presentation continues to play. The player has a further facility of allowing the author to restart the presentation from the same point as the previous time, so

that the author can easily replay the same part while changing different aspects with the editor.

The CMIF player is an important aspect of the authoring environment not only in terms of the way it plays a presentation, but also in the way the player system is able to communicate with the individual editors in the environment. Part of its strength lies in its invisibility—the player appears to be straightforward and simple yet it is intricately tied in with the rest of the environment. Much of its power is in the scheduling software which is mostly invisible to the author. A more detailed description of the player is given in [RJMB93].

6.4.5 CMIF player

The CMIF player is part of the authoring environment to the extent that an author needs to be able to check how the presentation will look to the end-user. Although this thesis concentrates on authoring support in a hypermedia environment, we feel the following points about the runtime environment are sufficiently important to be included.

Pre-arming of data

The CMIF player supports pre-arming for media items, that is, the player is able to fetch the data for the following item while the current item is still playing. This enables continuous playing without distracting gaps in the presentation. Pre-arm times for the media items are stored in the document from previous sessions. These, plus the timing information stored in the document, are used at runtime for calculating whether there is time to look ahead and fetch the data for the following items.

Pre-arming is currently implemented only for the next items in all the active presentations. The current implementation does not pre-arm two or more items per active channel, nor for links. It would be desirable to implement pre-arming on links, since, from the end-user's point of view, whether the next part of the presentation is in the currently playing scene or not it should arrive just as promptly on the screen. This would require the pre-arming of an inactive presentation, which is equivalent to starting up new invocations of the player for each link destination, with all the associated overhead.

Alternate data types

While the role of channels in an authoring environment is mainly that of allocating layout and style information to multiple atomic components, at runtime they are used for resource control. In Section 6.3.1 on atomic components we discussed the advantage of allowing alternate media items for different network loads or end-user machine specifications. This requires the player to make decisions at the atomic component level as to which content to use. A better solution, for the player environment, is that the decision be made at the channel level—the natural place for making resource decisions. The disadvantage for the author is that multiple atomic components have to be created. In the current environ-

ment this action can be carried out by copying existing atomic components then changing only the content referred to and the channel used. The advantage is that at runtime the player can switch channels on and off and ignore dependencies at the data layer.

6.5 Resources

The environment layer contains stores of information that, while outside the scope of an individual hypermedia document specification, are resources that are needed by a document and can be reused by multiple documents. These resources are data format, style, channel and attribute.

6.5.1 Data format

A data format resource is needed so that intrinsic spatial and temporal information can be deduced from the media item and so that a player can present the item. This information is stored in CMIFed in the list of channel types and not as a store of explicitly supported data formats. The system interprets the data format using the libraries called by the implementation of the channels. The author is thus supported by the system recognizing standard formats and is not required to specify the format for every media item in the presentation.

6.5.2 Styles

Style information allows the display characteristics of events to be described. CMIFed supports the assignment of styles to channels and atomic components. The override priority is that the channel specifies the default style and the atomic component can override it. This can be seen in the atomic component information dialog, Fig. 6.3(b), where, by clicking on the name of a property, the default value is shown. CMIFed does not support styles for composite components. This means that copying a structure and changing only the style of the components requires changing the style of all the individual components rather than specifying an override for the whole composite.

Styles are not separate from channels, so that styles have to be copied among channels rather than shared. A style facility allowing styles to be assigned to channels would be a low-cost improvement.

CMIFed supports only media-specific styles. A separate style facility could also support the definition of media-independent styles with variations for different media types. For example a "preferred anchor" style could be an outline within text and a pop-up within an image.

Discussion

We are not aware of any system other than CMIFed that currently allows the specification of styles for multimedia, or that allows styles to be assigned via channels. We consider one of CMIFed's important contributions to be the introduction of styles for multimedia.

6.5.3 Channels

CMIFed supports the use of channels. These can have associated data format, layout and style information but no explicit semantic attributes. A channel can only be used within one document, although a document can itself consist of multiple presentations. CMIFed supports the creation, copying, pasting, editing and deletion of channels.

A channels can be a layout channel, allowing several channels to belong to one layout. This allows hierarchical specifi cation of position and styles. These can be overridden in an individual channel. Their size and position can also be defi ned in terms of the layout channel.

An apparent restriction is that channels have a fi xed size and position. Channels are not only introduced as a high-level style/layout specifi cation, but, at least as importantly, as virtual resources that a playback environment can use for pre-calculating screen usage, or audio channel allocation. From an author's perspective, it is useful to vary the size or position of the channel while a media item is playing within it. It is questionable, however, whether this is acceptable for the channel's virtual resource function. The player would have to carry out far more assessment work on the predictable resource usage, and substantially more processing power would be needed, e.g., if an image channel changes size then the image would have to be rescaled continuously at runtime.

Edit channel

A CMIFed channel has associated with it a data format, a position, size and styles.

- CMIFed allows the use of different image data formats within an image channel. A video channel is data format specifi c. An audio channel can interpret two audio data formats.
- The channel position and size is defi ned in terms of a layout channel or a window. A channel cannot base its height, width or position on a sibling channel.
- In CMIFed a channel can have associated style information, specifi ed as part of the channel information, depending on the media type.

Assign Channel

In the atomic component editor the channel associated with the component is selectable from a list of channels, illustrated in Fig. 6.3(b) " Channel". CMIFed does not check the data format of the media item when the author selects a channel, so is not able to offer an appropriate subset of channels to choose from. This is because it is the channel software which is able to interpret the data format. A warning could, however, be given to the author that the channel does not recognise the data type. If an inappropriate channel is chosen then the player cannot display the media item.

CMIFed does not have the notion of the current layout and so is not able to restrict the set of channels from which to choose. A low cost improvement to the

current editor would be to support this facility. The editor could further support the author in showing which channels from the layout have already been assigned in the current composite component.

Discussion

In the current editor, when a channel similar to an existing channel is required, then a new channel has to be specified. A more appropriate mechanism would be to allow all channel properties to be shareable and overrideable, in particular layout and styles. We recommend that such an inheritance principle be implemented.

In summary, we believe one of the major contributions of the CMIFed work is the introduction of channels for separating out resource issues from the rest of the concerns of the author. Channels may appear cumbersome in the CMIFed environment. Their advantages become more apparent, however, when the media items included in a presentation are distributed among different (possibly specialist) servers and when the same source document is to be played on end-user platforms with widely differing display and performance characteristics. The support for channels in CMIFed is further advanced than in many other playback environments, although there are still areas which can be improved, such as separating out styles from channels.

6.5.4 Attributes

CMIFed does not support attributes explicitly in any way. There is a comments field for atomic and composite components which can be used by the author for informally naming, or describing objects, but this is not used by any other part of the system.

Discussion

Although there is no attribute support currently in CMIFed, this is one of the directions in which we plan to take the work. This would involve finding appropriate ways of labelling both media items and components in the presentation, and being able to use these for creating presentations more automatically. Related work is ongoing in this field[BFMR97], [WeWi96] and we have carried out some initial work in this direction, see for example [HaBu95a], [BuRL91].

6.6 Summary and Conclusions

The CMIFed authoring environment supports the majority of the document model described by the AHM. The parts of the model supported are detailed in Appendix 2 of the thesis. In providing editing support for the model, the environment satisfies the majority of the requirements, as set out in Chapter 5, for a hypermedia authoring system.

6.6.1 Data layer

CMIFed, in conjunction with facilities provided by the operating system, supports most of the authoring requirements for the data layer. The exceptions are selecting part of a media item for non-image data, and an explicit list of data formats which can be interpreted by the CMIF environment.

6.6.2 Component layer

Atomic

We consider one of CMIFed's important contributions to be the introduction of atomic components to a multimedia authoring environment. By separating out data dependencies from the rest of the environment they allow an author to edit the presentation in a uniform manner and allow multiple re-use of media items.

Advantages of the environment are that duration and size do not have to be specified explicitly per atomic component. Duration is derived from the surrounding structure, where possible, and layout is inherited from the channel hierarchy.

An improvement which would require a large amount of implementation effort is:

- full support throughout the environment for transitions.

Straightforward to implement enhancements to the current environment are:

- the use of markers for defining audio anchor values;
- the assignment of specified durations for continuous media items, using a low cost cut-off or repeat scaling method;
- anchor-specific presentation specifications;
- font scaling by specifying it in relation to the size of the channel.

Potential improvements to the environment, but which we feel would not be worth the implementation effort, are:

- anchors external to the marked-up data for text;
- overlapping text and image anchors;
- specified absolute durations using the options of playing faster and stretching.

Composite

CMIFed supports a number of composition authoring requirements directly and supports other requirements via channels. The combination of the hierarchy and channel views has proved to be a successful way of defining structure and timing relations at a high level while allowing the specification of more detailed timing relations. The creation of structure within the hierarchy view can sometimes become a burden to the author, although this may be a consequence of the complexity of the presentation the author is creating rather than an artefact of the hierarchy view.

CMIFed permits the copying of composite components for inclusion in other parts of a presentation, but not literal re-use. For re-use to be of benefit within

the environment, the assignment of styles, spatial or timing information to composites would also need to be implemented. To include a re-use feature would require the resolution of a number of functionality issues as well as the design of an appropriate user interface.

An improvement which would require a large amount of implementation effort is:

- support for the creation of synchronization arcs to and from composite components. This would require the design and implementation of an interface where both time and structure are visualized.

A straightforward to implement enhancement to the current environment is:

- the assignment of a duration to a composite component by scaling the durations of the descendant atomic components by an appropriate factor, using a low-cost cut-off or repeat scaling method for continuous media.

A potential improvement to the environment, but which we feel would not be worth the implementation effort until support for attributes has been implemented, is:

- support for composite anchors.

Link

Link support in CMIFed has not been a major focus of the work. The CMIFed link editor serves both as a link manager and link component editor. The interface supports the navigation and selection of components and anchors as the predominant means of interaction. While bidirectional links are supported the editor, we do not believe this is essential as part of a hypermedia environment. Major omissions are transitions on links and improving the editing environment for link components.

An improvement which would require a large amount of implementation effort is:

- full support for source and destination contexts of a link;
- support for the explicit selection of source and destination contexts.

Straightforward to implement enhancements to the current environment are:

- anchor highlight style for selected and destination anchors;
- support for specifying the correct document structure to give the desired link behaviour;
- allow the author to specify an ancestor choice component as the source context;
- support for links with a single destination component including multiple choice components, where the author should be able to specify a destination anchor for each choice component.

Potential improvements to the current environment, but which we feel would not be worth the implementation effort, are:

- showing the source and destination components of a link simultaneously in the hierarchy view, requiring two different foci is required. The current link

editor compensates for this to some extent by allowing the direct selection of a component from the channel or hierarchy view focus;
- giving the end-user control over the activation of each separate presentation, where it has to be clear which player control is coupled to which active presentation;
- support for styles for specifiers.

6.6.3 Document layer

Timing

The channel view shows a timeline based view of each child of a choice component, and the author is able to navigate among the children. Within each channel view an author is able to define synchronization constraints between atomic components. We consider one of CMIFed's important contributions to be the approach taken to deriving the durations of components from the composition structure and intrinsic durations of continuous media items.

Potential improvements to the current environment, but which we feel would not be worth the implementation effort, are:
- making a visual distinction between intrinsic and author-specified durations in the channel view;
- support for abrupt or gradual changes in tempo;
- support for higher-level specification of synchronization arcs;
- a zoom-in facility for the timeline of the channel view.

Spatial layout

Layout support is provided in CMIFed using channels, which bring many advantages, for example allowing the end-user to resize the presentation at runtime. Potential disadvantages, such as less flexibility of layout, proved to be a problem only in the case of trying to specify position in relation to the content of an image. Sizes and positions of channels can be viewed in the player.

Straightforward to implement enhancements to the current environment are:
- show only the positions of channels within the same layout.

Potential improvements to the environment, but which we feel would not be worth the implementation effort, are:
- allow the size of an object to change with time;
- visualize an overview of the layout throughout a presentation;
- support the creation and visualization of paths;
- enable the positioning of a component with respect to an anchor within another atomic component;
- support the automatic assignment of channels based on the structure of the presentation;
- allow the comparison of layouts used in different parts of a presentation.

Link management
CMIFed does not provide complete link management facilities. For more robust handling of links a complete link management system would need to be implemented.

Presentation Control
The CMIF player is an important aspect of the authoring environment both in terms of the way it plays a presentation and in the way the player system communicates closely with the editors in the environment. The player gives the end-user control over the flow of the presentation as a whole and also over the individual channels. Much of its power lies in the scheduling software, e.g. the algorithm for pre-arming the next media items on the currently active channels. The channel view gives an overview of the components which are playing and those which are being pre-armed. An important improvement would be the implementation of pre-arming for links.

Potential improvements to the environment, but which we feel would not be worth the implementation effort, are:

- fast-forwarding and reversing the presentation;
- some means of visualizing which presentations, as specified in the hierarchy view, are active.

6.6.4 Resource

Data format
CMIFed interprets the data format using the libraries called by the implementation of the channels, and does not describe it in the document format.

Style
We consider one of CMIFed's important contributions to be the introduction of styles for multimedia.

Straightforward to implement enhancements to the current environment are:

- a separate facility for defining styles which can also be applied to channels;
- support for styles in composite components.

Potential improvements to the current environment, but which we feel would not be worth the implementation effort, are:

- support for media independent styles.

Channel
We believe one of CMIFed's major contributions is the introduction of channels for separating out resource issues from the rest of the concerns of the author. The advantages of channels become more apparent where the media items included in a presentation are distributed among different servers and where the same source document is to be played on end-user platforms with widely differing display and performance characteristics. The support for channels in CMIFed is much further advanced than in any other current playback environment that we are aware of.

CMIFed supports the use of channels as multimedia styles and for spatial layout. Channels can be layout channels, allowing several channels to belong to one layout, allowing hierarchical specifi cation of position. Styles can also be inherited from layout channels.

Channels have a fi xed size and position, which at fi rst sight appears to be a restriction. Since, however, one of the functions of channels is to assist in resource allocation, there needs to be some constancy so that a playback environment can pre-calculate the resource allocation.

Straightforward to implement enhancements to the current environment are:

- allow the selection of channels from a choice restricted to those belonging to one layout;
- carry out data format checking when a channel is assigned to an atomic component.

Potential improvements to the environment, but which we feel would not be worth the implementation effort, are:

- support for inheritance of layout as well as style properties among channels.

Attributes

Although there is no attribute support currently in CMIFed, this is one of the directions in which we plan to take the work. This would involve fi nding appropriate ways of labelling both media items and components in the presentation, and being able to use these for creating presentations more automatically.

An improvement which would require a large amount of implementation effort is:

- support for attributes on all components as part of a more automated approach to authoring

6.6.5 Conclusion

We summarised the functionality of CMIFed and classifi ed potential extensions to the current environment as being:

(a) worthy of implementation although they would require a large amount of effort,

(b) straightforward to implement and thus worth the implementation effort, or

(c) not worth the amount of effort in the current authoring environment.

6.6.6 Beyond CMIFed

The implementation of the CMIFed environment demonstrates that editing support can be supplied for a document model such as the AHM. We have shown that the model, although perhaps at fi rst sight complicated and unwieldycan be edited satisfactorily through a divide and conquer approach. Only a small number of model elements cannot be edited in the current environment because

of the user interface, in particular, the lack of an interface to creating synchronization arcs among composite components.

While we feel that the interfaces designed in the system are, apart from the reported omissions, functionally adequate, they do not necessarily present the most suitable user interface to an author. Questions can be asked on two levels—how does the environment fi t together as a whole, and how can the individual parts be improved. In particular, although the hierarchy view corresponds to a temporal structure editor and the channel view to a timeline, there may be better ways of integrating the different parts of the environment with each other and better ways of providing and visualizing the correspondances among the views. Other improvements lie in the area of interactivity of the interface, e.g. by allowing the author to manipulate the representations of the atomic components in the channel view directly, for example to change the duration.

In order to answer these questions, the system has to be used and assessed by a range of authors. This is currently being undertaken in the Chameleon project [Cham95], where a number of companies already engaged in the production of multimedia presentations are using the system and supplying their feedback to the designers of the system.

A Hypermedia Authoring Environment: CMIFed