Modelling and Authoring Hypermedia Documents

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam,
op gezag van de Rector Magnificus
prof.dr J.J.M. Franse
ten overstaan van een door het
college voor promoties ingestelde commissie
in het openbaar te verdedigen
in de Aula der Universiteit
op dindag 3 maart 1998 te 11.00 uur

door

Hazel Lynda Hardman

geboren te Glasgow

Promotor:	prof.dr ir A.W.M. Smeulders
Co-promotor:	dr D.C.A. Bulterman, CWI, Amsterdam
Promotiecommissie:	prof.dr P. Paolini, Politecnico di Milano, Italië dr R.H. Trigg, Xerox PARC, CA, Verenigde Staten prof.dr M.L. Kersten, Universiteit van Amsterdam prof. M.M. Chanowski, Universiteit van Amsterdam
Faculteit:	Wiskunde, Informatica, Natuurkunde en Sterrenkunde Universiteit van Amsterdam

The research reported in this thesis was funded by the following:
SAFE, DELTA project (P7061, D1014),
OWL, Office Workstations Limited, Edinburgh,
MAGUS, STW project PBTS of the Dutch Ministry of Economic Affairs,
STEM, Telematics project EN-1014,
CHAMELEON, ESPRIT-IV project 20597,
CWI, Centre for Mathematics and Computer Science, Amsterdam.

ISBN: 90-74795-93-5

Contents

1 Summary

Hypermedia presentations are documents which are not printed on paper but make use of a computer screen for their display. Fig. 1 shows an example hypermedia presentation. The example shows text, video, images and audio combined together into different presentations and includes choice points where a reader can select other presentations to view. We use the term *media items* for the pieces of text, video etc. and call the choice points *links*. When combining media items into a presentation, temporal relations among the items specify when each item should appear on the screen and for how long.

A hypermedia presentation can be generated at play-back time from an underlying document which specifies the various aspects of the presentation. To allow presentations to be played on different software systems, a model of the

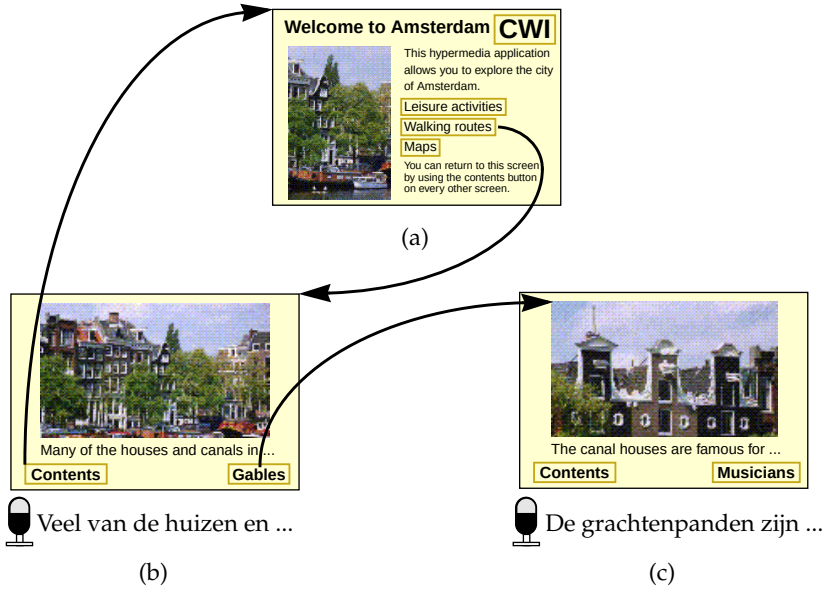


Figure 1. An example hypermedia presentation

underlying document is needed. An analogy is with word processors, where authors are able to use the system of their choice and the documents can be transferred among systems, because the underlying models are similar. Another advantage of having an explicit model is that documents can be processed for other reasons, such as creating multiple versions for different end-user platforms, or changing the visual styles of the document. It is this degree of compatibility and processability which we seek to achieve with a model of hypermedia documents.

Given a model for hypermedia documents, an authoring system can be created to support the creation, editing and deletion of the constituent parts of documents.

This thesis first states the requirements and defines a document model for hypermedia. It then analyses the user interfaces in existing authoring systems for multimedia documents and goes on to state the requirements for a complete hypermedia authoring environment. Finally, it describes the CMIFed authoring system, implemented by members of the CWI multimedia group.

1.1 Requirements for a model for hypermedia

Looking again at the example in Fig. 1 we can deduce a number of the requirements for a hypermedia document model.

Firstly, there are a number of media items. It should be possible to specify which items are included in the presentation, and which part of the item is to be displayed. The latter is useful for avoiding duplication of similar media items. The data type of each media item also needs to be known so that the playback system can interpret it.

Each media item is displayed at some position in the window and with a specific size. For example, the heading in Fig. 1(a) is at the top left of the scene and does not overlap with the CWI logo to its right. Timing information is also needed, e.g. to specify that all four items in Fig. 1(a) are to be displayed at the same time. Each media item needs an associated start time and duration.

As well as information per media item, other information involving multiple media items is needed. For example, the scene in Fig. 1(a) is a collection of the four items on the screen. Structuring information such as this must also be included as part of a document model.

The choice points must also be specifiable. These include the information about where a reader can click on the screen (for example the lightly shaded boxes in Fig. 1) and what the destination is. Links in hypermedia become quite complex since the presentation consists of multiple items, of which some can be continuous media, such as video and audio. When the reader follows a link only some of the presentation may change—for example in Fig. 1 (b) and (c) the *Contents* text item remains unchanged when following the link from the *Gables* text item.

While the emphasis of the document model is to ensure that a presentation can be reproduced on the basis of the stored document, it can also be used to support the retrieval of different parts of a document. Each media item, or part of a media item, can represent a real-world object or concept. The document model can be extended to include media independent descriptions of the information contained in the media items.

These are thus the requirements for a document model for hypermedia. In the thesis we show that existing models of hypertext (in particular the Dexter model) and multimedia (the CMIF model) are insuffi cient for describing all the required aspects for hypermedia. In particular, the following aspects are missing from existing models.

- The specifi cation of which part of a presentation is affected on following a link. For example, should the whole presentation be replaced by the destination of the link, should only one part be replaced, or should the destination appear in addition to the original presentation?
- The inclusion of style information describing how the source of the link should transform into the destination of the link. For example, it is not visually appealing if the presentation stops playing, the screen goes blank and the destination only starts playing a few seconds later. A feeling of continuity is given if the presentation continues playing but fades out gradually while the new presentation fades in.
- The inclusion of media-independent descriptions for parts of media items. For example, allowing an author to associate the notion of "house" with parts of the images in Fig. 1.

We conclude that a new model for hypermedia documents is required.

1.2 A hypermedia document model

In chapter 3 we defi ne the Amsterdam Hypermedia Model (AHM) which satisfi es the requirements derived in chapter 2.

The main elements of the model are the atomic, composite and link components and the channel. These incorporate other sub-elements in order to provide the required expressiveness for the complete model.

Atomic component

An atomic component collects together all the properties that can be associated with a single media item, including a reference to the media item. The properties are: duration, spatial information, style, and media-independent descriptions. An atomic component also allows the specifi cation of parts of a media item, termed anchors, which can be used as the start and end-points of links. An anchor requires a data-dependent specifi cation of part of the media item and can also have associated properties.

Summary

Composite component
A composite component allows the grouping of a number of other components into an element that can be treated in the same way as an atomic component. There are two types of composite component: temporal and atemporal. A temporal composite allows the specification of temporal relations among its children, thus forming a continuous presentation. An atemporal composite allows the grouping of presentations which have no pre-specified temporal relations among one another. This allows the creation of independent scenes through which a reader is able to navigate. A temporal composite includes synchronization arcs which record the temporal relations. An atemporal composite includes activation information recording which of the child presentations should be made active when the parent is activated, e.g. by following a link to it.

Link component
A link component specifies the source and destination components for following a link in hypermedia. The component consists of a number of specifiers, each of which can act as a source and/or destination of a link. Each specifier has a reference to an anchor which may be the place the reader clicks, or may be highlighted when at the destination of a link. The specifier also denotes which parts of the presentation are associated with the anchor, e.g. the atomic component containing the anchor or a composite component representing the complete scene.

Channel
The model also includes a channel element. This brings together spatial, style and data format information in a form that can be re-used by multiple atomic components.

By deriving the model from an example presentation we demonstrate that the parts of the model are necessary for describing a hypermedia presentation. They must also be shown to be sufficient, i.e. that the documents from a broad selection of systems conform to the model. We show that the AHM is able to describe the models implicit in a selection of existing hypertext, multimedia and hypermedia systems. We therefore conclude that the AHM is a comprehensive, yet not overly complex, model for hypermedia documents.

1.3 Authoring paradigms

Different approaches to authoring multimedia already exist. We analyse a selection of existing systems and categorise the authoring approaches into a number of paradigms which represent the underlying models presented in the user interface.

We analyse the paradigms to determine their suitability for different parts of the authoring task, namely creating narrative structure, temporal information,

spatial layout and links among individual presentations. Structure based systems are more suited to editing the structure of a presentation, and, where the structure reflects the temporal structure, are also useful for editing the presentation's timing. Timeline based systems are more suited to showing the timing throughout a presentation and the timing relationships among parts of a presentation. Flowchart and script based systems are best at specifying more generalised interaction, where the flowchart has a better user interface. None of the paradigms is particularly suitable for editing layout or for creating links, although the structure-based paradigm allows the different parts of the link to be specified.

We conclude that each paradigm is most suited to a particular editing task, that no single paradigm is sufficient for covering all editing aspects of a hypermedia presentation, and that several interfaces within a unified environment are required.

1.4 Requirements for authoring hypermedia

Given the AHM, we specify the functional requirements for an authoring system supporting the document model. To maintain an overview of the authoring environment it is divided into four layers: data, resource, component and document.

The data layer contains the media items themselves, thus shielding the other layers from data format dependencies.

The resource layer contains the resources used for the different aspects of the document, for example a data format resource required for interpreting the data, style information for fonts, media-independent descriptions dependent on an application domain, and layout information. The importance of including the resources as a separate layer is that each can be replaced by a similar resource while leaving the document structure itself unchanged. This allows multiple presentations to be generated from the same underlying document structure, e.g., layout can be tailored to specific output environments.

The component layer stores the document components and is supported in an authoring environment by allowing the editing of individual components. For example, a picture becomes part of a presentation when it is specified where, when and for how long it should appear on the screen.

Temporal and spatial layout play a particularly important role in hypermedia presentations, and these aspects have to be coordinated among multiple elements. The document layer allows the editing of these aspects and communicates the information to the component and resource layers. As well as stating the requirements for the document layer, illustrations of potential user interfaces are given for aspects such as editing temporal and spatial information. In particular, timeline illustrations are given for showing temporal constraints, changes in tempo, and navigating the presentation timeline.

Summary

1.5 A hypermedia authoring environment—CMIFed

Having specified the requirements for an authoring system, we describe CMIFed which was implemented for creating hypermedia documents. The importance of CMIFed from the perspective of this thesis is to show that the majority of the stated authoring requirements can indeed be implemented. For the requirements that were not implemented we are able to state what the reason was. This may have been unacceptable implementation effort, or that with hindsight they were not actually needed. CMIFed also illustrates ways of visualizing some of the aspects of the requirements, in particular temporal and spatial relationships among components.

1.6 Applications of work

The work described in this thesis has contributed to and benefited from two major international collaborations. Our model heavily influenced the development of the Synchronized Multimedia Integration Language (SMIL). This was developed by the World Wide Web Consortium working group on Synchronized Multimedia, which includes members from the CWI multimedia group. The SMIL language is being developed as a vendor-neutral multimedia document description language. Browsers are currently being developed which will enable SMIL documents to be played over the Web.

Work on the authoring system has continued as part of the Chameleon project, ESPRIT-IV Project 20597. The project benefited in two ways. Firstly it had immediate access to a hypermedia authoring environment, CMIFed, whose functionality, broadly speaking, satisfied the requirements derived in this thesis. Secondly, one of the goals of the Chameleon project is to allow the creation of a single source document which can then be (semi-)automatically adapted for playback on a range of end-user platforms. The CMIFed document format is based on the AHM, which allows it to be translated with relative ease to other formats. Target formats currently being implemented in the Chameleon project are MHEG-5 and SMIL.

1 Introduction

Multimedia

The bird sings a song.
The artist paints a picture.
They say the same things.

Hypermedia

The grasshopper sings.
She jumps to another leaf.
So much more to see!

Humankind has recorded information for tens of thousands of years [Clot95]. Information was captured initially as images before developing to a symbolic system of text. Time-based media became common-place only one hundred years ago through the development of technologies such as film and gramophone records. The introduction of computer technology, around fifty years ago, had an impact as a new storage medium, but had little consequence for the media types used in digital encodings of documents. Instead, the computer provided a convenient means for creating and storing familiar media. Initial support was for textual documents, followed by image support. Computing power is now sufficient that support can also be provided for time-based media, such as film and audio.

The ability to record information brings with it the ability to comment on and make explicit relationships between pieces of information. When adding commentary in paper-based textual documents, a scholar requires to refer to both the original work and to the interpretations offered by other scholars. The ability to reference existing material has also been implemented for computer-based documents [EnEn68]. This gives the advantage that references to other computer-based material can be followed directly. Computer support provides in this case not only convenience but a qualitative improvement with far-reaching consequences. The most familiar example of computer support for direct referencing among documents is the World Wide Web [BuRL91].

The media used for recording information are no longer bound to their carrier technology, such as paper, film or audio tape, but become unified in a digital environment. This allows the different media to become part of a larger whole. An important example is the creation of on-line time-based presentations [HoSA89]. These allow the specification of pieces of information along with when and where they are to appear on the screen.

Computer-based presentations are thus currently able to include text, image and time-based media elements, synchronization among elements and referencing among presentations. In order to create and view these presentations, tools for building and playing them are required. Before embarking on the development of such tools, however, consensus needs to be reached on the underlying form of the presentations, that is, to define the underlying document model for these presentations.

This thesis has two goals. The first is to define a document model which allows the description of presentations incorporating multiple media types, including time-based media, synchronization among elements and referencing among presentations. The second is to determine the authoring system requirements for the model.

Terminology

We use the term *presentation* to refer to the runtime behaviour of the information units presented to the user. We use the term *document* to refer to a static description of the presentation which can be stored. The document can include static and dynamic information units and may be presented via different display media such as video or paper. A document may be a declarative or procedural description of the presentation. In order to display a presentation based on a particular document description both the document, plus a player environment that is able to interpret the document description, are required.

Hypertext, multimedia and hypermedia are common terms with no consensus on their definitions, so we give the definitions as used throughout this thesis. *Hypertext* is a description of the referencing information among self-contained units of information. A *hypertext document* is a collection of information units and referencing information, called links, Fig. 1.1(a). A *hypertext presentation* is the runtime manifestation of one or more hypertext documents with which a reader can interact allowing the reader to navigate through the hypertext document. The information units in a hypertext document may include media types other than text. A commonly used term for this is hypermedia.

Multimedia is a collection of multiple units of information that are constrained by temporal synchronization relationships. A *multimedia document* is a collection of information units and associated synchronization information, Fig. 1.1(b). A *multimedia presentation* is the runtime manifestation of a multimedia document. A reader can interact with a multimedia presentation by, e.g., starting or pausing the presentation.

We use the term *hypermedia document* to denote a collection of information units along with referencing and synchronization information, Fig. 1.1(c). A hypermedia document is thus a collection of multimedia documents along with referencing information. A *hypermedia presentation* is the runtime manifestation of one or more hypermedia documents. A reader can interact with a hypermedia

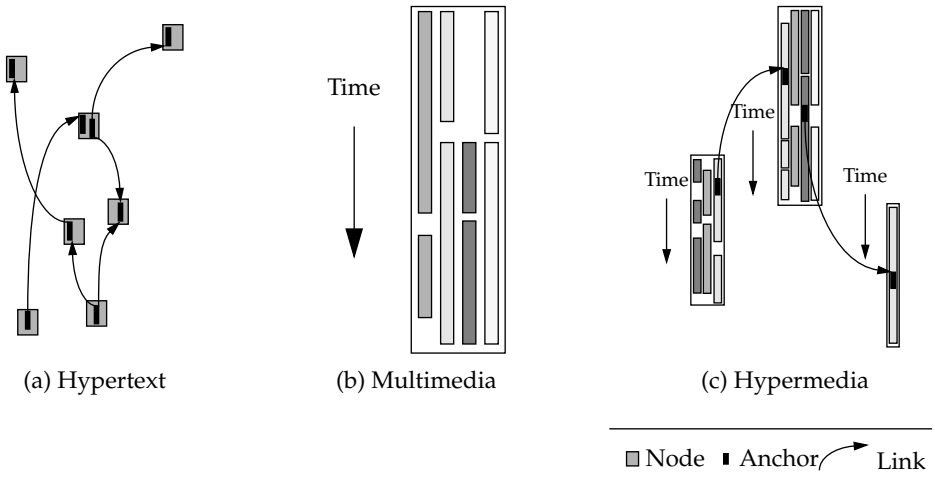


Figure 1.1. Hypertext, multimedia and hypermedia documents

presentation either as a multimedia presentation, by starting or pausing a multimedia presentation, or as a hypertext presentation, by navigating through the information units.

Scope of the thesis

A requirement for a hypermedia model is that it is sufficiently expressive that the same document can be presented on different platforms while preserving the author's original intentions. On the other hand, when a model becomes more complex there is a danger that it becomes too difficult to specify for any particular presentation, with the consequence that an authoring system becomes cumbersome to use. In the extreme case, a hypermedia presentation can be programmed directly in a non-specialist programming language which provides flexibility but minimal reuse and an unsupportive authoring environment. A simple model, supported by easy-to-use tools, is in turn too restrictive to allow the specification of all the required aspects of the presentation. The goal is to find a pragmatic trade-off between these two extremes.

In order to derive the requirements for a model of hypermedia documents we consider a typical presentation illustrating the aspects of a presentation we wish to model. This combines aspects of both multimedia and hypertext in a generalised model. Multiple media types, including moving images, sound, images and text, should be eligible for inclusion in a document.

Once we have defined a model we investigate how authoring support can be provided. Our goal is to integrate multiple items of differing media types within a single document, so we do not discuss editing environments for individual media, e.g. word processors or sound editors. We assume, instead, that the media items have been created or that an author has access to the appropriate tools for creating them.

Existing authoring systems for multimedia do not use a uniform approach to creating presentations. We analyse the different approaches in order to gain insight into the utility of these approaches in a more complete hypermedia authoring environment.

Using this analysis and the hypermedia document model definition we construct a list of authoring system requirements for hypermedia documents. Having derived the required functionality, a system interface also needs to be specified. This requires a complete interface design and falls outside the scope of the thesis. We do, however, provide examples of existing user interfaces for parts of the document model. The thesis includes a description of the authoring system CMIFed [RJMB93] to show that the majority of the requirements derived in the thesis can be implemented within a single environment.

Structure of the thesis

Chapter 2 presents a simple example hypermedia presentation and describes the aspects that need to be recorded in a hypermedia document model to allow the presentation to be reproduced. This ensures that a model based on the requirements will contain necessary features for describing a presentation. In the second part of the chapter we compare these requirements with existing candidate models for hypermedia. We conclude that these models are insufficient and that a new model needs to be defined. In Chapter 3 we define a model for hypermedia based on the requirements. We show that the model is sufficient as a model for hypermedia by describing the document models implicit in a selection of hypertext, multimedia and hypermedia systems in terms of the model. We conclude that the proposed model is a suitable model for hypermedia documents.

Chapter 4 analyses the authoring paradigms embodied in existing authoring systems. We conclude that each paradigm is particularly suited for at most one of the multiple authoring tasks in creating a hypermedia presentation. In Chapter 5, we formulate and argue the authoring requirements for a hypermedia document based on a careful decomposition of the authoring process by way of the model defined in the first part of the thesis. We concentrate in particular on the aspects of temporal and spatial layout within a presentation and of creating links among presentations, since these are the defining characteristics for hypermedia presentations. In Chapter 6 we describe the CMIFed environment which satisfies the majority of the authoring requirements.

2 Requirements for a Hypermedia Document Model

Using an example of a typical hypermedia presentation as a base, we discuss the features of the presentation and the corresponding requirements for a document model that would be able to specify the presentation. Features discussed are: media items forming the basis of the presentation, composition, temporal and spatial layout, and style information. We also discuss activation state and information retrieval and their consequences for a document model. We describe existing models for hypertext and multimedia and conclude that these are insufficient as a model for hypermedia. We state the list of requirements for a comprehensive hypermedia document model.
This chapter is based on work presented in [HaBu97], [HaBR94], [HaBR93b], and [HaBR93a].

2.1 Introduction

In this chapter we first specify the requirements for a hypermedia document model and then evaluate existing hypertext and multimedia models in terms of these requirements. We establish that existing candidate hypermedia models do not meet the requirements developed in this chapter. We conclude that a new model is necessary for describing hypermedia documents.

Our requirements for a hypermedia document model concern the following features of a hypermedia presentation:

- information about an individual media item,
- specification of parts of a media item,
- additional information associated with an instance of a media item when included in a presentation,
- composition of instances,
- specification of relationships among instances and compositions,
- temporal and spatial layout for instances and compositions,
- styles applicable to document elements,
- semantic information associated with instances and compositions and
- information for runtime presentation and control.

In order to clarify our discussion of the requirements for a document model we classify the requirements according to the layers of a hypermedia document

processing system: characteristics of media items used in the presentation; specification of structural relationships among elements of the document; and runtime characteristics of the presentation. The first two layers have relationships between the media items and the document elements, and the second two layers have relationships between the elements and their final presentation. These layers are taken from the Dexter model [HaSc94], which, while developed for a model for hypertext, are sufficiently broad to be applicable to hypermedia document processing. Dexter terms these three layers: within-component, storage, and runtime, Fig. 2.1. The *within-component layer* stores the details of the content and internal structure of the different media items used in a presentation; the *storage layer*[1] describes the document structure; and the *runtime layer* is where user interaction is handled. The Dexter term given to the interface between the within-component layer and the storage layer is *anchoring*. The Dexter term given to the interface between the storage layer and the runtime layer is *presentation specifications*. We describe the Dexter model in more detail in Section 2.3.1.

A number of the requirements we consider in the course of this chapter are similar to those implicitly satisfied by the Dexter model or by aspects of HyTime.

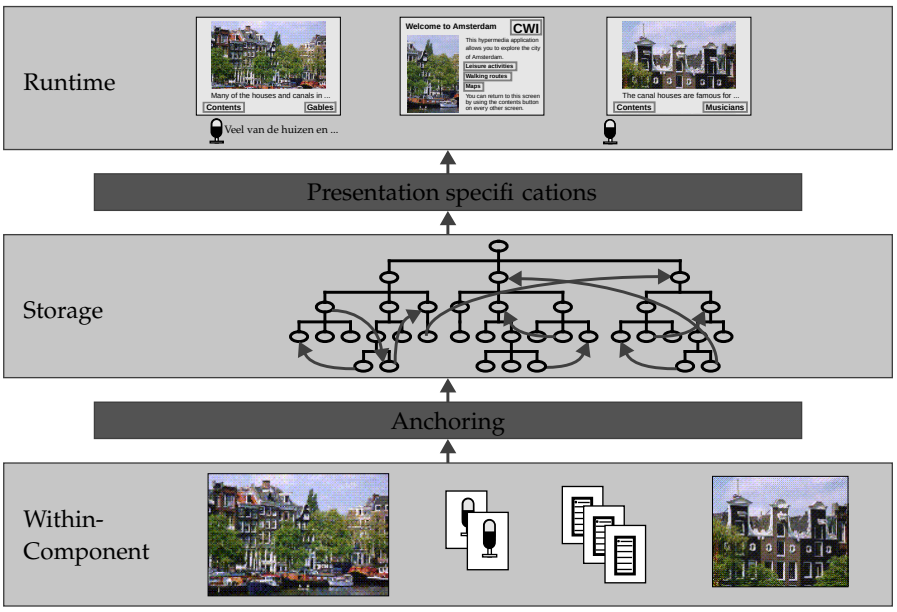


Figure 2.1. Layers of the Dexter model used for classifying requirements

1. The Dexter term storage layer is slightly misleading, since information for all of the layers needs to be stored.

The requirements tend to be expressed in Dexter terms, and so we refer explicitly to HyTime constructs where appropriate. These are included as footnotes.

We motivate our requirements for a hypermedia document model using a simple example of a hypermedia presentation. This provides an intuitive introduction to the requirements. The example presentation incorporates synchronized discrete and continuous media and allows the reader to make selections within the information. We deliberately choose a small example to provide a lower bound on the requirements necessary for describing a hypermedia presentation. Whether our requirements are suffi cient for describing presentations created by a broad range of systems is an empirical question. We postpone this question until Chapter 3, where we show that our proposed document model is able to describe the presentations created by a range of existing systems. In order to unify the various discussions in the following sub-sections, we refer back to this example throughout this chapter.

The presentation shown in Fig. 2.2 illustrates three fragments from a tour of the city of Amsterdam. The top fragment, (a), is analogous to a table of contents and provides the reader with a description of the tour and a number of options from which to select. This continues to be displayed until the reader selects another scene to be displayed. One of these is a description of a walking route through the city, highlighting a number of places of interest found on the tour. Two places of interest are shown in Fig. 2.2(b) and (c).

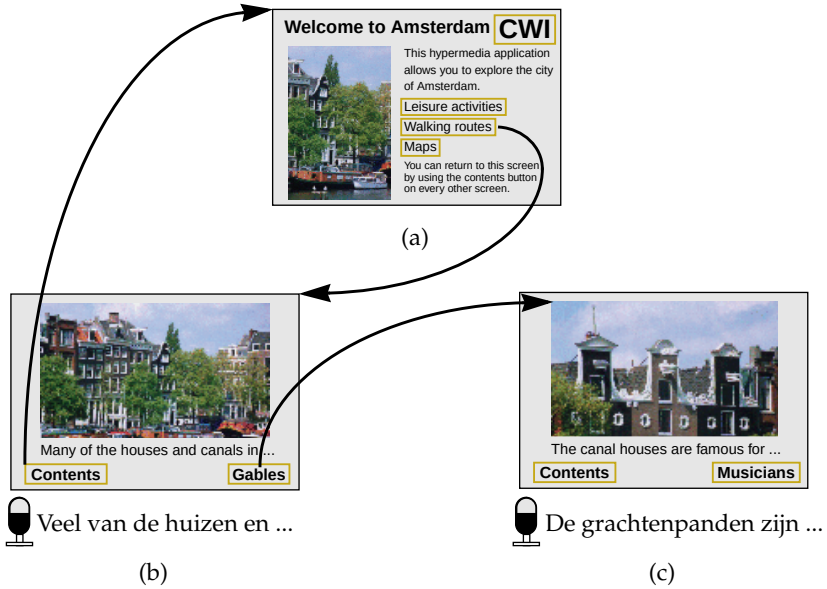


Figure 2.2. An example hypermedia presentation

For each scene to be displayed, the player requires:

- the data for each of the media items,
- the starting time and duration of each instance of a media item in the presentation and
- its extent and position on the screen.

For example, in Fig. 2.2(a) a specification for the position of the heading text is required, as well as its typeface and size. Thus the requirement for the document model is that for each media item there needs to be information about its position, the start time and duration of its display, plus style information.

Further specifications are needed for supporting navigation among presentations. For example, in Fig. 2.2(a) the initial opening scene is playing and at some unspecified point in time the reader selects to go to the walking route in (b). At this point the opening scene fades out from the screen and the first section of the walking route fades in. The action of making the selection requires information for specifying the following:

- where the reader is able to make a selection, e.g., the three boxed phrases in (a);
- where each selection leads, e.g., "Walking routes" in (a) leads to the scene in (b), and "Gables" in (b) leads to (c); and
- how the transition should be made from the scene that was playing to the newly selected scene, e.g., when going from (a) to (b) nothing remains of the presentation in (a), whereas when going from (b) to (c) the "Contents" text is common. The scene in (a) fades into the scene in (b), whereas the scene in (b) does a "wipe left" to the scene in (c).

The requirement for the document model is that for each selection information is required for specifying a part of a media item, for associating a destination with the point of selection, for specifying the scope of the presentation affected on following the link and for describing the special effect associated with making the selection.

This simple example illustrates a number of features of a hypermedia presentation that need to be specified as part of a hypermedia document model. In the following section we go into greater detail for each of these.

This chapter is structured as follows. Section 2.2 states the requirements for a hypermedia document model and discusses each of these in detail. Section 2.3 describes existing models for hypertext and multimedia, compares the requirements with the existing models and lists the limitations of the models as models for hypermedia. Section 2.4 concludes that we need a new model for hypermedia. The specification of a hypermedia model is given in the following chapter.

Dexter layers	Model features	Section
Within-component layer	Media items	2.2.1 Within-Component layer: Media items
Anchoring	Reference to part of media item	2.2.2 Anchoring
Storage layer	Properties associated with instance of media item	2.2.3.1 Instance of media item
	Composition	2.2.3.2 Composition of instances, 2.2.3.3 Composition of anchors
	Linking	2.2.3.4 Linking
	Semantic attributes	2.2.3.5 Semantic attributes
Presentation Specifi cations	Temporal layout	2.2.4.1 Temporal layout
	Spatial layout	2.2.4.2 Spatial layout
	Styles: media item, anchor, transition	2.2.4.3 Styles
	Initial activation state	2.2.4.4 Activation state
Runtime layer	Temporal flow	2.2.5.1 Temporal control
	Spatial layout	2.2.5.2 Spatial control
	Navigation (activation changes)	2.2.5.3 Navigation control

TABLE 2.1. Document model features for hypermedia

2.2 Requirements for a hypermedia document model

We use the example in the previous section as a starting point for specifying the requirements for a hypermedia presentation. We structure the discussion using the Dexter layers and go through the layers in a bottom-up manner. We fi rst describe characteristics of the media items, and then go through the problems of including and combining these into a presentation. Table 2.1 gives a summary of the document features and where they are discussed in this section. Throughout this section, the emerging document model requirements are stated and summarised in tabular form. These smaller tables are collected together at the end of the section in Table 2.13 which provides a complete summary of the document model requirements.

2.2.1 Within-Component layer: Media items

A media item contains the data that is presented to the reader and as such is the basis of a presentation. We defi ne the*media item* as an amount of data that can be retrieved as one object from a store of data objects—although not necessarily a small amount. Media items can be of different media types. For example in Fig. 2.2(a), the screen shown consists of four media items—a video of a canal scene, the CWI logo, a heading and a longer text item. Although a multimedia presentation is built up from heterogeneous pieces, it is perceived by the reader

as a continuous whole. The goal is to integrate different media types into the presentation while retaining media type independence of the document model.

We first discuss the nature of a number of common media types and then discuss their dimensionality.

Media types

For the purpose of combining different media types within a single presentation, we define the characteristics of the four basic media types: text, image, audio and video.

- *Text* is an ordered linear sequence of two-dimensional characters, e.g. English language text, braille, Chinese characters.
- *An image* is a static two-dimensional, visual representation, e.g. a real world image such as a drawing or a photo, or a symbolic representation such as a graph.
- *Audio* is a continuous audible medium, e.g. speech, music, or sound-effects.
- *Video* is a continuous sequence of moving images, e.g. continuous real-world images, animation or any sequence of still images that is intended to be perceived as a unity.

A number of media types, such as text and vector graphics, can of themselves be structured. Text is a special case in that the internal structure of the media item can be expressed using the components of it, e.g. HTML [Ragg97] or Postscript®[Adob90].

A media item may consist of a single medium, such as those just described, or a composite medium such as interleaved video and audio.

Temporal and spatial dimensionality

We distinguish two categories of media–continuous and non-continuous, or discrete. Continuous media have a temporal dimension intrinsic to the media item itself, e.g. audio and video. Non-continuous media have no intrinsic temporal dimension, e.g. text and images.

Fig. 2.3 shows representations of the four basic types of media items in a three-dimensional space.

- Text requires two spatial dimensions for display, Fig. 2.3(a). The aspect ratio of the display area is relatively unimportant, since lines can be broken at various positions without altering the semantics of the message.
- Images also require two spatial dimensions, Fig. 2.3(a), but the aspect ratio is important for aesthetic reasons as well as for accurate representation of real-world objects.
- Video requires two dimensions of space plus time to be displayed, Fig. 2.3(b). Video can be regarded as a sequence of images, where each image is displayed at a particular time. The aspect ratio is thus important.
- Audio is a continuous medium and has no spatial dimensions, Fig. 2.3(c).

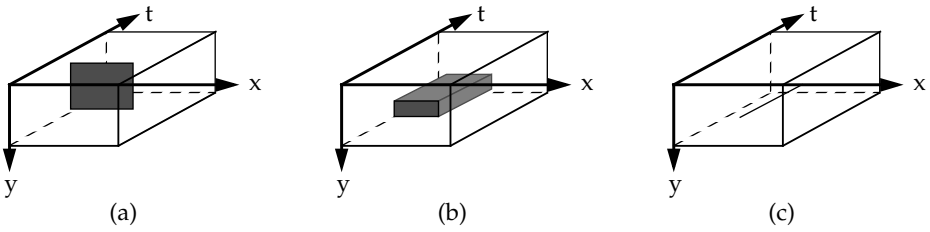
While we portray the presentation here as taking place in two spatial dimensions plus a time dimension, three spatial dimensions plus a time dimension is

also a possibility (although representations on two dimensional paper would be difficult to interpret). For example, with virtual reality applications the media items are three dimensional objects in a three dimensional space whose position, extent and orientation can change with time. The other aspects of the model addressed in this section are similarly not restricted to three dimensions, but can be illustrated within them.

When a media item is incorporated into a multimedia presentation it is displayed on the screen for some duration or played through an audio device. Thus, in order to incorporate a media item within a multimedia presentation knowledge of its temporal and spatial dimensions is required[2].

Generated media items

Other data types that should be includable in a presentation are outputs from external programs or processes, for example the video signal from a camera pointing at an outside scene, the reading from a monitoring device in a chemical plant or power station, or computer synthesized music. For live feeds the data can be treated as having a pre-specified spatial extent but with an indefinite duration. Alternatively, data generated on-the-fly from an external program may be of known spatial extent and duration. For example, financial results generated from a market simulation are displayed as a graph in a presentation, [HaRB95]. The requirement for including live data, program code or other generated media items in a presentation is the same as for the standard data types—that the spatial and temporal dimensionality be known beforehand. While it is also useful to know the spatial and temporal extents, where the duration may be indefinite, this is not a requirement.



(a) Text and graphics–spatial but no intrinsic temporal dimension.
(b) Video and animation–spatial and temporal dimensions.
(c) Audio–temporal but no spatial dimension.

Figure 2.3. Spatio-temporal dimensions of media types

2. These coordinate spaces are similar to the concept of finite coordinate spaces (FCS) in HyTime [ISO97b].

Dexter layers	Model features	Model requirements
Within-component layer	Media items	temporal and spatial dimensions

TABLE 2.2. Model requirements for the within-component layer

2.2.2 Anchoring

The content included within a presentation does not necessarily have to be an entire media item, but might be a reference to part of it. For example the picture of the gables as used in Fig. 2.2(c) is only a part of the original picture, shown at the bottom of Fig. 2.1. This allows multiple use of the same data without the need for extra storage. The content can be given by a reference to the stored media item and a media-dependent specifi cation of a part of it. Examples for other media types are the following. In the case of text the media item may be a complete book, where only a section is required. In audio, for example, a selection from a music item may last a number of seconds, but may also be only one track for the length of the complete item. A video segment might be a combination of temporal and spatial cropping operations, where a number of frames are selected from the complete sequence (cropping in time) and only a part of the image is shown (cropping in space).

To allow a reader to make a selection to go to another scene there needs to be information on the screen denoting where the reader can make a selection, and what the expected information at the destination will be. For example, in Fig. 2.2(a) there are three choices within the text item plus one on the CWI logo. These selections need to be specifi ed within the document.

In order to create a synchronization relationship with a point or an interval in a continuous media item a means of specifying the point within the media item is needed. For example, in Fig. 2.2(b) and (c) there is a spoken commentary. At selected points in the commentary the subtitles change. When exactly within the spoken commentary the subtitles should change needs to be described.

In all three cases, a data-dependent specifi cation of part of the media item is required. We term this the anchor value [HaSc94].

Dexter layers	Model features	Model requirements
Anchoring	Reference to part of media item	data-dependent specifi cation of part of media item

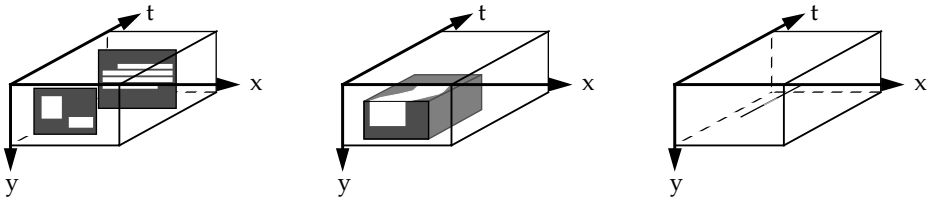
TABLE 2.3. Model requirements for anchoring

We give examples of anchor values in different media types. A graphical interpretation is given for text, image, video and audio anchors in Fig. 2.4.

- A *text anchor value* normally specifi es a sequence of characters within a text item, Fig. 2.4(a). This may be given as a character offset and length of a text string, but, to withstand possible editing of the text, this should also specify keywords such as the beginning, subject and end of the text fragment, along

with extra context information about where the fragment can be found. In some cases a text query may be the most appropriate, for example stating what the text should be about rather than which letters should be part of the text.

- An *image anchor value* specifies an area in a pixel-based image, where most systems implement the area as being rectangular, Fig. 2.4(a), although there is no theoretical restriction on its shape and any contour could be defined to specify its extent—allowing objects in the image to be traced out. In a vector graphic image an anchor may refer to any object (single or grouped) in the image. The point is that the internal specification is data format dependent and can refer to anything appropriate to the data format. Similarly an anchor could be specified in a 3D graphic as an object within the graphic, for example a house in a virtual reality landscape.
- A *video anchor value* may be specified as a sequence of frames, as is used in a number of systems [Davi93], [HjMi94]. This allows the user to select at most one link to follow to another presentation from any frame in the video. A more desirable approach is to specify the area on the screen for the extent of the frame sequence [SmZh94]. This allows several choices for each frame, but the choices do not vary during the sequence. A complete description is given when each area is specified per frame in the sequence, [BuKW94]. This allows moving or changing objects in the video to be followed, so that clicking on an object becomes a more natural option, illustrated in Fig. 2.4(b). For a live video feed, objects within the video image could be designated as anchor values, but real-time image recognition would be needed to implement them.
- An *audio anchor value* can be specified as a time partitioning of the audio, illustrated in Fig. 2.4(c), or it might be a timed extent of one musical voice, for example a number of bars of violin solo. The problem starts when the reader tries to interact with the audio item, since with the normal mode of



(a) Graphics anchor, e.g. area; text anchor, e.g. text string.
(b) Video anchor, e.g. area changing with time.
(c) Audio anchor, e.g. temporal extent, or in music a temporal extent within an instrument or voice.

Figure 2.4. Anchors

interaction with hypermedia presentations (clicking an object on the screen) there is nothing tangible with which to interact–although links *to* audio items remain possible. An example of interacting with " hyperspeech" is given in [Aron91]. Here, although no anchors were used in the application, the authors suggest using a Doppler shift effect to suggest that the listener is approaching, or passing, a hyperspeech branch. The anchor value would be a portion of the audio data, and the Doppler effect the presentation style.

Anchor values can be specified for other media types, although there is less agreement on their form. For example, for a simulation program distinct states of the program could be used as anchor values.

The anchor value is specified in terms of the data format of the media item it refers to. Just as the media item has its own intrinsic spatial and temporal dimensions, the anchor value inherits these dimensions. Not only is the dimensionality the same, but the position and extent of the anchor value is defined in terms of the temporal and spatial axes defined by the media item. The consequence of this is that an anchor value cannot be scaled without also scaling the media item it refers to.

Another requirement is for the synchronization of anchors in a static medium with those in a continuous medium. Examples include a piece of text that has an accompanying spoken commentary, or a music score with its associated performance. To show the correspondence between the anchor values, an anchor value in the static medium should be able to be highlighted for the duration of the corresponding anchor value in the continuous medium.

2.2.3 Storage layer

The storage layer is where the media independent structure of the presentation is stored. While the media item is the basis of a presentation, information needs to be associated with each instance of a media item in a presentation. An *instance* is the inclusion of a media item in a presentation. An instance requires the specification of the media item along with properties associated with its inclusion. In order to create scenes, collections of instances need to be specifiable. In order to make selections among scenes there needs to be a construct to store the relationship. In this section we discuss the requirements for an instance, composition, and for making selections among scenes.

2.2.3.1 Instance of media item

When including a media item in a presentation information in addition to the data for the media item is required. This may be a specification of which part of the media item is displayed in the presentation, as discussed in the previous section on anchoring, or other types of information. For example, in Fig. 2.2(a) the position and extent of each of the media items, the font typeface and style for the text items and an appropriate background colour is needed. In Fig. 2.2(b) the

subtitles change with time, so that some specification of when each subtitle should be displayed is also required.

The list of properties required for specifying an instance of a media item are:

- a reference to the storage of the data object for the media item and its data format;
- a specification of the part of the media item which is to be presented;
- duration and start time of the media item;
- extent and position of the media item.

These properties are required, since without some knowledge of their value the player is unable to display the instance at the appropriate position and time in the presentation. The list of properties that may be associated with an instance of a media item are:

- aspect ratio;
- orientation;
- Z-order, i.e. the front to back ordering of media items;
- style;
- start points for choices of destination;
- semantic information for finding media items.

The duration or extent of a media item may be intrinsic to the media type of the item. If this is the case, then the duration or extent of the item as incorporated in the presentation may require some other value. This may be in terms of a scale factor or an absolute value. Start time and position cannot be specified for a media item in isolation from the rest of the presentation but need to be specified in relation to other media items or with respect to the presentation as a whole. As a consequence, these cannot be stored as properties along with the media item, but as part of the structure of the presentation. We discuss temporal specifications further in Section 2.2.4.1.

The aspect ratio, orientation and Z-order are applicable to screen-based media. We discuss these further in Section 2.2.4.2.

The style of presentation of the media item is dependent on the media type, and includes font size for text, or a colour map for images. We discuss style in more detail in Section 2.2.4.3.

In order to incorporate selection points within a presentation, there needs to be some way of recording where these start points should be. We discussed this in Section 2.2.2 on anchoring.

While not obligatory for the display of a hypermedia presentation, some means for finding relevant parts of a presentation is useful for authors creating presentations out of pre-existing parts, or for readers looking for specific pieces of information. Attaching semantic information to a particular use of a media item allows that use to be categorised, indexed and searched upon. We discuss this in more detail in Section 2.2.3.5 on attributes.

In a model for hypermedia the encapsulation of a media item in an instance should be expressible along with a means for associating other required and optional properties of that use of the media item as part of the presentation.

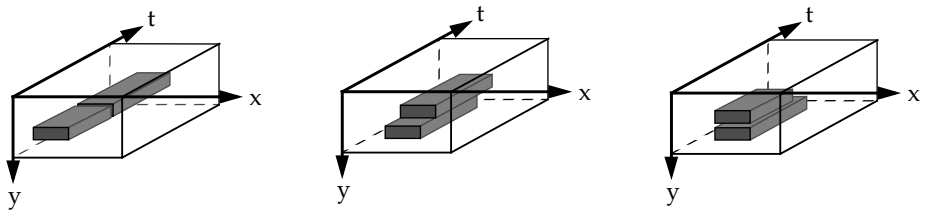
Dexter layers	Model features	Model requirements
Storage layer	Instance of media item	reference to (part of) media item, data format, duration, start time, extent, position, aspect ratio, orientation, Z-order, style (media item, anchor, transition), start points for links, semantic attributes

TABLE 2.4. Model requirements for an instance of a media item

2.2.3.2 Composition of instances

One of the distinguishing characteristics of a multimedia presentation is that it consists of a number of instances of media items combined together into an integrated presentation, for example, each of the three scenes shown in Fig. 2.2. A hypermedia model thus requires a composition mechanism for defining which instances are included in the presentation and how they are combined together, in particular their temporal and spatial relations. We name this type of composition space/time-dependent composition. *Space/time-dependent composition* specifies a number of children, along with temporal and spatial information relating the children. This type of composition allows the merging of smaller presentations into a larger presentation, Fig. 2.5. The timing of the children and the spatial aspects have to be specified in terms of the same coordinate axes[3].

Space/time-dependent composition alone is insufficient for describing a hypermedia application, since an application can consist of more than one multimedia presentation, as illustrated by the separate scenes shown in Fig. 2.2. These separate presentations need to be associated together in some way, but without merging them into a larger presentation. A second composition mechanism is thus required which allows the composition of multiple presentations. We name



(a) Sequential (b) Intermediate (c) Parallel

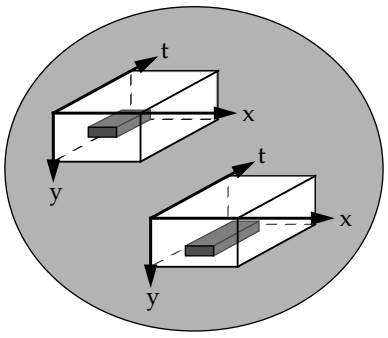
Figure 2.5. Space/time-dependent composition

3. Space/time dependent composition of instances is similar to the HyTime [ISO97b] placement of events within the same finite coordinate space.

this type of composition space/time-independent composition. *Space/time-independent composition* requires the specification of a number of children, but with no other temporal or spatial information. This type of composition allows the inclusion of separate presentations in a hypermedia document, Fig. 2.6. If any one presentation is playing then the only method of playing another is to follow a link to it. If no link exists to a presentation then it can never be played, since it is not part of the flow of some larger presentation. There is no intrinsic temporal relation among the presentations, and it cannot be predicted exactly when, or indeed if, the reader will follow a link to any one of them.

An example of space/time-independent composition is where multiple presentations can be played simultaneously. For example, when the reader selects the CWI logo in Fig. 2.2(a) a spoken commentary is given about the institute. The start time of the spoken commentary is not bound to the presentation already playing, but is conditional on the reader following the link from the logo. The time bases of the two presentations are independent within the document specification, but the presentations are played simultaneously when the reader follows the link.

Two other types of composition can also be identified: temporal composition and spatial composition. *Temporal composition* is time-dependent but space-independent; *spatial composition* is space-dependent but time independent. Two important categories of temporal composition are parallel and sequential composition ([Acke94], [HaRe94], [HaRB93]), Fig. 2.5(a) and (c). In the parallel case, the temporal relation is that the instances start together; in the sequential case that one instance starts when the previous finishes. These are sometimes treated as fundamental divisions, although they are two extremes of temporal composition. An intermediate case is that one instance starts and at some time later, but before the first finishes, the second one starts, Fig. 2.5(b). An example of using



A composite element, denoted by the shaded ellipse, contains two sub-elements aligned along independent coordinate axes.

Figure 2.6. Space/time-independent composition

temporal composition with spatial independence is where a presentation is playing and after a pre-specified time a second presentation starts to play in another window. The reader is able to move the positions of either window independently.

An example using spatial composition is where a presentation is built up of several subscenes. A number of items are able to remain on the screen (e.g. in Fig. 2.2(b) a link back to the contents screen) while the reader selects the different subscenes (e.g. in Fig. 2.2(b) a picture with spoken commentary and a text item linking to the next subscene). There is no pre-specified temporal relation between the items that remain on the screen and those playing in the subscenes. The spatial relation is, however, fixed. Another example of spatial composition is described in [Grøn94], termed a table top composite, where the spatial relations among the child elements are recorded.

In a model for hypermedia temporal, spatial and space/time-independent composition are essential, fundamental structuring mechanisms and should be expressible. Space/time-dependent composition is an alternative to separate temporal and spatial composition mechanisms. These requirements are summarised in Table 2.5.

2.2.3.3 Composition of anchors

A multimedia presentation is, by definition, assembled from a number of items of different media. These items, however, may contain illustrations of the same concept. For example, in Fig. 2.2(b) the word "houses" in the subtitle, the houses in the picture and the spoken Dutch word "huizen" within the commentary all illustrate the concept "houses". A means is required for grouping the different expressions of the concept as a single element. This would allow the creation of a single relationship from the "houses" concept rather than forcing the creation of multiple relationships with otherwise identical properties.

The grouping of anchors with similar concepts is orthogonal to the temporal/atemporal structuring of the presentation. For example, the three anchors described above happen to be displayed at the same time, but there may be other illustrations of the concept "houses" which occur at other points in the presentation. These may also form part of the single conceptual anchor.

Note that composition of anchors is similar in spirit to generic links in Microcosm [HaDH96].

The model requirement is that parts of different media items expressing a single concept should be able to be treated as a single element.

Dexter layers	Model features	Model requirements
Storage layer	Composition	temporal and spatial composition, space/time independent composition, grouping of anchors into composite anchors

TABLE 2.5. Model requirements for composition

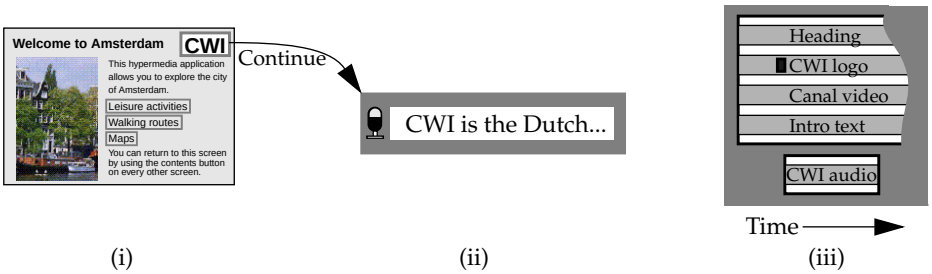
2.2.3.4 Linking

While viewing an interactive on-line presentation, a reader should be able to make choices as to which information is displayed. For example, in Fig. 2.2(a) there are four highlighted areas from which the reader can choose. When one of these is selected the destination of the link is displayed. For example, by clicking on " Walking routes" in (a), the introduction is replaced by the scene in (b). In the case of hypermedia, where multiple items are playing simultaneously, they may not all be replaced, but some subset of them. For example, by selecting " Gables" in (b), not all of the scene is replaced by the scene in (c)—the " Contents" text remains playing. In this case, the scope of the information associated with the link is a part of the original presentation. We call this scope specification the *link context* [HaBR93b], [NaNa93]. The *source context* is the information associated with the source of the link and the *destination context* that associated with the destination of the link. There may be multiple source and destination contexts, but for the sake of simplicity we discuss the situation of only one source and one destination context.

Not only does the source context need to be specified, but also what should happen to the source context when the reader follows the link. There are three options which we illustrate in Fig. 2.7. The source context can remain playing, so that the destination context is a new, independent, presentation which is started up, Fig. 2.7(a). The source context can remain on the screen but pause, (b). The source context can be cleared from the screen and the destination context played on its own, (c). Note that when following a link the source and destination contexts become part of the same virtual temporal composition when the reader makes a link selection. The information needed to specify the link transition is the same as that needed when grouping elements to become part of the same temporal composition.

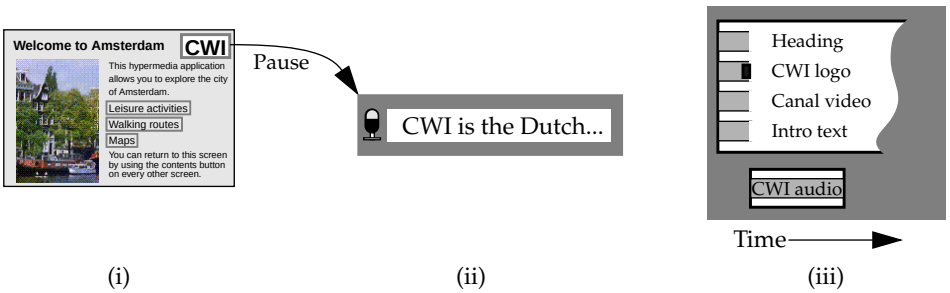
When a reader follows a link from the source to the destination context, this should not introduce a break in the flow of the presentation, but should be a smooth transition from one scene to the next. The action of going from the source to the destination context we call the *link transition*, [SFHS91]. A link transition has a duration and an associated style. The duration is the length of time it takes to play the transition. The style is a special effect that can be applied to the source context as it changes into the destination context, for example, effects such as " wipe-left", " zoom-out", " dissolve", or sound effects. The transition might even be a sequence in its own right. For example, if a user chooses to visit Amsterdam from a map of the earth, actioning the link doesn't make the presentation jump to a presentation about Amsterdam, but increases the scale of the earth gradually then dissolves into the new presentation.

When the link has been followed and the reader arrives at the destination, it can be useful to highlight a particular object at the end of the link. This can be achieved by specifying a destination anchor which can be highlighted in a man-



(a) Continuing the source context

While the scene in (i) is playing the reader clicks on the CWI logo. The scene continues while a spoken commentary (ii) is delivered. The same interaction is seen from a time-based point of view in (iii).



(b) Pausing the source context

While the scene in (i) is playing the reader clicks on the CWI logo. The scene pauses and a spoken commentary (ii) is delivered. The same interaction is seen from a time-based point of view in (iii).

(c) Replacing the source context

The reader clicks on the *Walking route* text and the scene is replaced by the first scene of the walking route. The same action is seen from a time-based point of view in (iii).

Legend: Playing item Reader clicks on anchor Paused item

Figure 2.7. Runtime behaviour of links in hypermedia

ner similar to, though not necessarily the same as, highlighting the source anchor. For example, when going from the walking route scene in Fig. 2.2(b) to the introductory screen in Fig. 2.2(a) the "Walking routes" anchor marker could be flashed once to indicate that is where the reader just came from.

In summary, when specifying a link in hypermedia the following is required:

- a source anchor that the reader can select in order to choose the destination;
- a specification of how much of the displayed presentation belongs to the source link context;
- a description of how the source context is transformed into the destination context, and whether it remains playing;
- a specification of the destination link context.
- A destination anchor may also be associated with the link to emphasize a particular part of the destination context.

Dexter layers	Model features	Model requirements
Storage layer	Linking	source and destination anchor, source and destination context, transition (duration and special effect) change in activation state change in playing/paused state

TABLE 2.6. Model requirements for linking

2.2.3.5 Semantic attributes

As more and more distributed sources of multimedia become available, some way is needed of labelling the information stored in these sources so that it can be found again. A further requirement is that presentations are needed for different circumstances, such as reader information requirements, display platform or available network bandwidth. Rather than creating these presentations individually, authoring effort can be spared by generating them from an underlying representation. For both information retrieval and for more automated authoring, some connection with the semantic content of the presentation has to be made.

One means of merging a multimedia presentation with a knowledge representation is to associate semantic labels, which we call *attributes*, from a knowledge representation with parts of the document structure. This is analogous to labelling a book with classifications from a library catalogue. A hypermedia document model should not define what the semantic attributes should be, but should provide hooks for attaching classification information. Alternatively semantic information may be associated with a multimedia presentation by having a knowledge structure refer to media items expressing a particular concept. This is analogous to a library catalogue listing the books it holds in each category. In both cases it is a many to many mapping, where each book or media

item can be associated with multiple categories or concepts, and each category or concept can be associated with multiple books or media items.

While the library classifi cation example does not go further than the book as a unit of classifi cation, in the case of hypermedia the labelling should be carried out for single media items, for collections of media items and for parts of media items [NaNa93], in particular for larger media types such as video [BuKW94].

By labelling media items with semantic attributes, fragments of presentations can be found that correspond with a reader's information need. Having retrieved the appropriate fragments, a larger presentation can be generated from them. For example Davis [Davi93], generates sequences of video from a store of labelled video clips and Worring et al. [WBHT97], specify the design of a system for generating hypermedia presentations on the basis of semantic labelling. Alternatively, the presentation can be generated top down from a knowledge-based description, e.g. André et al. [Andr96] generate multimedia presentations on the basis of domain information.

Although a hypermedia document model should not specify the form of the attributes, it is useful to give an illustration of the way we expect attributes to be used.

- *Anchors*
  Anchors in instances correspond to the basic semantic objects which can be seen by the reader and described by the author. There will no doubt be higher level, or more abstract, concepts involved in the presentation, but these are more diffi cult to point at directly. For example, a bicycle and wheel shown in a video are labelled with the semantic attributes " bicycle" and " wheel".
- *Instance*
  For an instance it is likely that there is a collection of attributes already associated with anchors of the instance. There need be no extra attributes associated with the instance, but there may be a higher-level abstraction associated with it. For example, an instance referring to an image with two anchors labelled " funny hat"and " cake"may have the attribute " birthday party".
- *Composition*
  Similarly, for a composition it is likely that there is a collection of attributes already associated with the descendants of the composition. An attribute associated with the composition may then be a higher-level, or more abstract, term.
- *Composite anchor*
  For a composite anchor, there is a collection of attributes similar to the collection for anchors in an instance, although the same attribute may be expressed in different media. For example, the word " bicycle"in a text item has the attribute " bicycle"and a picture of a bicycle has the same attribute.

The composite anchor referring to both anchors still has the same attribute " bicycle".

- *Link*
  The attributes associated with a link are slightly different from those associated with the other objects. A link specifies a relationship among objects, and thus the attributes would be expected to reflect this. For example, a link from an anchor with the attribute " wheel" to one with " bicycle" would have the attribute " is-part-of". A link from an anchor with the attribute " mountain bike" to the same " bicycle" anchor would have the attribute " is-a-type-of".

In conclusion, a hypermedia document model should allow semantic information to be associated with anchor values, instances, compositions, composite anchors and links.

Dexter layers	Model features	Model requirements
Storage layer	Semantic attributes	associate with anchors, instances, compositions, composite anchors and links

TABLE 2.7. Model requirements for semantic attributes

2.2.4 Presentation specifications

The presentation specifications form the interface between the storage layer and the runtime layer. Presentation specifications for multimedia include temporal and spatial layout and style. Temporal layout is fundamental to multimedia and is the specification of when an instance is presented and how long it remains playing. Spatial layout is the specification of the position and extent of the instance on the screen. Style information includes the choice of colour, font, etc. for the various media items used within the presentation, the signalling of anchors and transition styles.

2.2.4.1 Temporal layout

Temporal layout is the determining characteristic of multimedia, in the same way that links form the basis of hypertext. In this sense, temporal properties take on a much greater importance than a small section of a hypermedia model. We make explicit where these temporal relations fit in with other relations in such a model, and give an overview of the types of temporal information that can be specified. The six subsections address: defining a time axis in relation to which instances can be played, the start time, duration and temporal scaling of an instance, temporal relations among instances and the temporal relation associated with a link transition.

Specification of time axis

Including an instance within a multimedia presentation requires specifying when it should be played in relation to some implicit or explicit time axis. A

common example of an explicit time axis is the timeline, Fig. 2.8(a), as used in Director [Macr97] and the Integrator [SFHS91]. A time axis can be defi ned implicitly when durations of instances are known and these are grouped together with known temporal relations among them. A number of systems use this method to derive a time axis: e.g. Eventor [ENKY94], Mbuild [HaRe94] and CMIFed [HaRB93].

In either case the rate of traversal along the timeline can be varied, in a similar way that music can change its tempo when being performed.

The requirement for a hypermedia model is that a time axis should exist, whether it is specifi ed explicitly or is implicit.

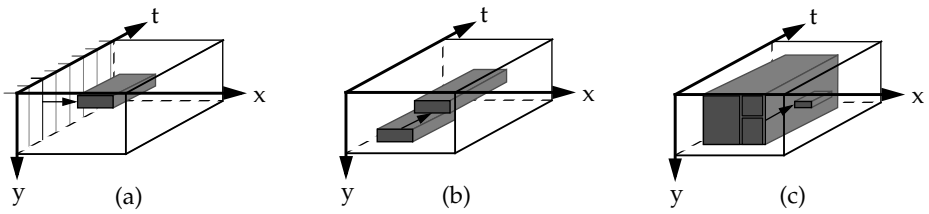
Start time of instance

The start time of an instance can be given in a number of ways. The most common method is to defi ne an instance's start time relative to a timeline, illustrated in Fig. 2.8(a). Examples of authoring systems using this approach are the Integrator [SFHS91] and Director [Macr97]. A second method is to specify the start time with respect to another instance in the presentation, illustrated in Fig. 2.8(b). This is supported, for example, by Firefly [BuZe93a] and Eventor [ENKY94]. A third possibility is to specify it with respect to a composition, illustrated in Fig. 2.8(c) and implemented in MET^{++} [Acke94].

The requirement for a hypermedia model is that the start time for an instance is expressible. This can either be specifi ed explicitly or can be calculated from relations with other parts of the document structure.

Duration of instance

As well as specifying when an instance should begin, its duration also needs to be known. This may be specifi ed explicitly or implicitly or derived from relations with other instances. For example, a video has an intrinsic duration associated with its media item. An image, however, needs to be assigned a duration. This may be explicitly assigned or derived from the surrounding presentation. An example of a derived duration is where the image is displayed when a spo-



(a) Start time is specifi ed with respect to a timeline.
(b) Start time is specifi ed with respect to another instance.
(c) Start time is specifi ed with respect to a space/time-dependent composite.

Figure 2.8. Ways of specifying start time

ken commentary begins and remains until the commentary has finished. In order to achieve this type of derived duration, media items need to have the property that they can be scaled along the time axis.

An anchor of a media item may also have an associated duration. The start or end time may be specified by a point in time, e.g. a frame number in a video, or "5 seconds after the start" of an image, or "the beginning of the word gables" in a spoken commentary. The duration can be given by an interval, e.g. a number of frames in a video, "between 3 and 5 seconds" for an image, or the time it takes to say the word gables in a spoken commentary. Alternatively, the duration may be specified using begin and end times, e.g. "from the beginning of the word distinctive to the end of the word gables" in a spoken commentary.

The requirement for a document model is that the duration of the instance is known or stated as unpredictable. This can be stated explicitly or deduced from relations with other parts of the document structure.

Temporal scaling of an instance

In order to satisfy temporal constraints that derive the duration of an instance, or satisfy constraints, individual media items need to be scaled in the temporal dimension. If the duration is to become longer then the instance can either be played slower or can be repeated until the specified duration is reached. If the duration is to become shorter, then the instance can be played faster or can be cut short.[4]

Another form of scaling, called temporal glue, [HaRe94] and [BuZe93a], allows variable length delays to be inserted into a document, so that when a media item's duration is changed, the other constraints remain satisfied.

Temporal scaling can be specified explicitly for atomic and composite instances in the MET^{++} system, [Acke94], and is derived from the document structure in the CMIFed system, [HaRB95]. Temporal scaling should be expressible within a hypermedia document model, along with an indication of how this should be achieved.

Temporal relations

Temporal relations among instances can be defined in terms of (a) whole media items, (b) parts of media items (anchors), or (c) groupings of items. Examples of each of these are: (a) a video starts simultaneously with its audio track; (b) each word in a subtitle is highlighted when it is spoken in an audio commentary (synchronization between anchors); (c) background music is required to start with the beginning of a sequence of slides and finish at the end of the sequence.

A commonly cited ([Bord92], [BuZe93b], [Erfl93]) categorisation of temporal constraints, put forward by Allen [Alle83], is given in Fig. 2.9. These allow all possible combinations of temporal relations between two instances to be expressed. More complex relations can be built up out of this set. Using the three examples

4. This is similar to extent reconciliation in HyTime [ISO97b].

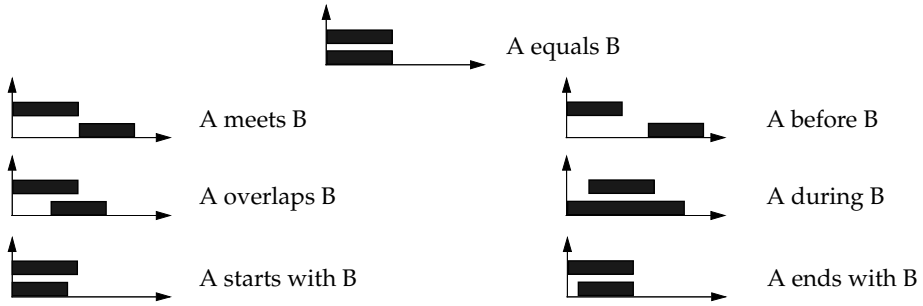
above we can illustrate the above three cases: (a) the video *starts with* the audio; (b) the highlighting of the word in the subtitle *equals* the duration of the spoken word; (c) the music *equals* the sequence of slides *meeting* each other.

Allen's relations make the assumption that the durations of the instances are known beforehand. If this is not the case, then a conditional action can be specified. For example, two media items are playing and when the first of them stops playing, which is unknown beforehand, the other one also stops [Erfl93]. Bordegoni [Bord92] gives a further categorisation of conditions as being deterministic or non-deterministic, and simple or complex conditions.

All of Allen's relations should be expressible between instances and compositions in a hypermedia document model. If, however, the duration of one of the instances in the relation is unpredictable, then a number of the relations can no longer be specified.[5]

Link transition temporal relation

When a link is interpreted as a navigation action, information needs to be specified for the presentation aspects. In particular, when moving from the source to the destination context, some specification needs to be given for the duration of



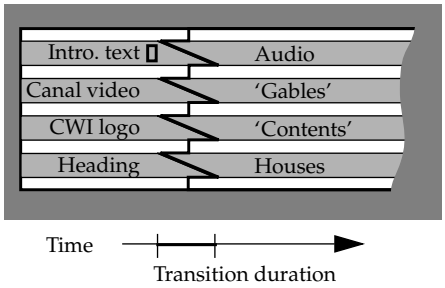
For each relation, apart from the *equals* relation, there is a corresponding B to A relation.

Figure 2.9. The Allen time relations

5. For the case that A has an unpredictable duration, the consequences for the Allen relations are the following.
A starts with B, B starts with A, B meets A, B before A: there is no change in the relationship.
A equals B, A during B, B during A, B overlaps A: the start time of B is known, but the relationship as stated cannot be guaranteed.
A meets B, A before B: the start time of B is unpredictable but can be scheduled when A ends.
A overlaps B, A ends with B, B ends with A: the start time of B cannot be scheduled since it is unknown when A will end.

Dexter layers	Model features	Model requirements
Presentation Specifi cations	Temporal layout	time axis, start time, duration, scaling, Allen's relations, link transition duration

TABLE 2.8. Model requirements for temporal layout



The source context fades out when the link is selected and the destination context fades in. The duration of the link transition is the time from the selection of the link to the full display of the destination context.

Figure 2.10. Temporal information in link transition

the action. For example, in the case of the source context fading out and the destination context fading in, the duration of the transition is the time from the start of the fade-out of the source context to the end of the fade-in of the destination context, Fig. 2.10. This could be more than a single duration, for example, where the source context fades out to black, there is a slight pause and then the destination context fades-in. The duration of the link transition should be expressible within a hypermedia document model.

2.2.4.2 Spatial layout

Spatial layout is an important characteristic of multimedia, defi ning where instances are displayed within the presentation. We give an overview of the types of spatial information that can be specifi ed. The eight subsections address: defi ning a space axis in relation to which instances can be positioned, the position, extent, spatial scaling, aspect ratio, orientation and Z-order of an instance, spatial relations among instances and the spatial relation associated with a link transition.

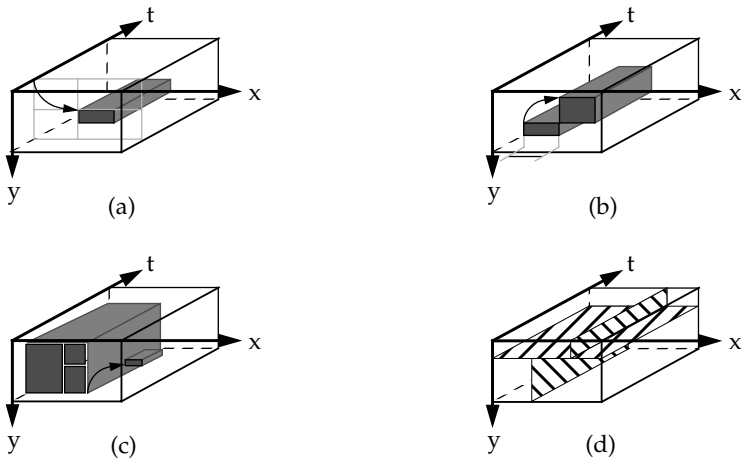
Specification of space axis

Including a media item within a multimedia presentation requires specifying where it should be played in relation to implicit or explicit space axes. The most obvious explicit space axes are the width and height of a computer screen or a

window on the screen. A space axis can be defi ned implicitly when the spatial extents of instances are known and these are grouped together with known spatial relations among them. We know of no systems that use this approach, however.

Position of instance
Spatial layout specifi cations of instances can be given in a number of ways. The most common method is to defi ne an instance's position per item and relative to the window (or screen) in which it will be displayed, illustrated in Fig. 2.11(a). Examples of authoring systems using this approach, are Eventor [ENKY94], Mbuild [HaRe94], Authorware and Director [Macr97]. A second method is to specify the relation with respect to another instance in the presentation, illustrated in Fig. 2.11(b). A third possibility is to specify the relation with respect to a composite instance, as is illustrated in Fig. 2.11(c). Neither of these last two methods is implemented in existing systems.

Another approach, not applied in the temporal case, is to divide the available space into predefi ned channels[HaRB93], illustrated in Fig. 2.11(d). This is similar to specifying the position with respect to a window, but the channels have their own spatial relations with respect to each other. When the position of a channel changes then all instances assigned to that channel also have their position changed. This has the same advantage as specifying position in relation to



(a) Position is specifi ed relative to a window.
(b) Position is specifi ed relative to another instance.
(c) Position is specifi ed relative to a composite instance.
(d) Channels predefi ne areas.

Figure 2.11. Ways of specifying position

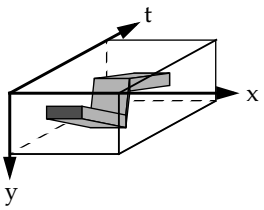


Figure 2.12. Position as a function of time.

an instance, but does not have the disadvantage that a relation has to be specified for every instance.

In all ways of defining the position of an instance, the position may vary with time, illustrated in Fig. 2.12. This is implemented in, for example, the MET++ [Acke94] and Eventor [ENKY94] systems.

A combination of the methods mentioned could be made, where a channel can move in time, another channel can be defined relative to the first, and an instance's position within a channel can change with time. The requirement for a hypermedia document model is that the position of an instance should be expressible.

Extent of instance

As well as specifying the position of an instance, its extent also needs to be known. This may be specified explicitly implicitly or derived from relations with other instances. For example, a video has an intrinsic extent associated with its data, or a text item can be assigned an explicit extent. An example of a derived spatial extent is where a subtitle is scaled to fill the width of its associated video. In order to achieve this type of derived extent, media items need to have the property that they can be scaled.

Extent can be defined in terms of a window or channel, relative to another instance, relative to a composite instance, and can change with time. The specification of extent is required in a hypermedia document model.

Spatial scaling of an instance

When incorporating an image or video shot into a presentation the original size of the item may not be appropriate in which case a scaling operation is required[6]. This may be an absolute size, such as "200 by 100 mm", or may be relative, such as "increase to 200%". If the extent is to become larger then the instance can be enlarged or repeated to fit the specified extent. If the extent is to become smaller then the instance can be shrunk or can be cropped. In the cases of enlarging or shrinking aspect ratio may need to be preserved, so that the specified extent may not be exactly achievable.

6. This is similar to extent reconciliation in HyTime [ISO97b].

This deformation may take place as a function of time. It may also be in relation to other instances, for example, "increase font size until heading fits above image". Spatial scaling should be expressible within a hypermedia document description.

Aspect ratio
Spatial scaling involves two dimensions, where the scaling of one dimension may differ from that of the other dimension. For images and video it may be important that the scaling factor is the same for both dimensions, i.e. that the aspect ratio be preserved. The model requirement is that the aspect ratio can be specified as needing to be preserved.

Orientation of instance
Given that there are at least two spatial dimensions within a presentation, an instance can be placed with a particular orientation at its position in the coordinate space. While this is omitted in most current multimedia systems, for applications involving three-dimensional media items, such as virtual reality, the orientation becomes more important. There is no equivalent of orientation for the temporal dimension since it is placement within a one-dimensional space. Orientation should be expressible within a hypermedia document description.

Z-order of instance
The Z-order of an instance is its position in a stack of instances occupying the same screen area. In other words, the instance with highest Z-order will be visible on the display. This is useful for occluding other instances, and also for transparent instances with graphical overlays. Z-order should be expressible within a hypermedia document description.

Spatial relations
Just as valid a method of specification, but to our knowledge not used in multimedia authoring systems, is to define the coordinates of a media item relative to those of some other media item, illustrated in Fig. 2.11(b). This would be useful for displaying, for example, subtitles next to the video they belong to, since if the position of the video is changed, the subtitles will remain in the same neighbourhood relative to the video.

Spatial relations among instances can be defined in terms of (a) whole media items, (b) parts of media items (anchors), or (c) groupings of items. Examples of each of these are: (a) subtitles are scaled to be the same width as a video; (b) a textual description is placed next to part of an image; (c) an image is scaled to form the background for a number of text elements.

Spatial constraints between instances parallel the temporal constraints shown in Fig. 2.9, but need to be combined in two dimensions. The three examples can be expressed as (a) the width of the subtitles *equals* the width of the video; (b) the text *meets* the image anchor horizontally and the image anchor *meets* the text ver-

tically; (c) the heights of the text items *before* each other are *during* the height of the image, the width of each text item is *during* the width of the image.

Link transition spatial relation
When a link is interpreted as a navigation action, information needs to be specified for the spatial relation of the two contexts. This may be in terms of the spatial information stored in the destination context, or a specified spatial relation in terms of the source context, thus defining a spatial relationship where none existed before. This brings the destination context into the same spatial coordinate system as the source context. The position or orientation of the destination context may change with time during the transition. The spatial relation of the link transition should be expressible within a hypermedia document model.

Dexter layers	Model features	Model requirements
Presentation Specifications	Spatial layout	space axis, position, possibly changing w.r.t. time, extent, scaling, orientation, Z-ordering, link transition spatial relation

TABLE 2.9. Model requirements for spatial layout

2.2.4.3 Styles
Style information is presentation information that applies to the display characteristics of document elements. We discuss here where style information may apply and what interpretations of this may be made. Specification of style information is required in a hypermedia document model.

Media item style
The style information for a media item specifies media-related display characteristics, relevant when the system is actually presenting the item to the user. Any one display characteristic may apply to multiple media types, for example background colour or anchor highlight colour, or to only one, for example font size or line spacing.

Anchor style
The style information for an anchor is needed for specifying how the visual, or audible, characteristics of an anchor can be specified so that a user is aware that an anchor is indicating the start of a link. Examples of anchor styles are:

- a border round the anchor value;
- for text items the use of different colour or styles such as underline or italic;
- the anchor value flashes;
- the anchor value changes colour when the mouse cursor is over it;
- the anchor value " pops-up", i.e. highlights and moves position slightly when the mouse cursor is over it;

- the anchor value enlarges slightly when the cursor is over it;
- the mouse cursor shape changes when it is over the anchor value.
- For an audio item the pitch of the voice changes when the anchor value is spoken,
- or a sound effect is given just before the beginning of the anchor value.

When a user selects an anchor to follow a link, there may also be style information associated with this action. For example the source anchor may highlight (to acknowledge that the action has been registered) before the destination of the link is displayed. The destination anchor may also be highlighted briefly to distinguish it from any other anchors present in the destination context. Examples of anchor highlight styles are:

- the thickness and/or colour of a border changes;
- for text items the style and/or colour changes;
- the anchor value flashes;
- the anchor value changes colour;
- the anchor value " pops-up"briefly;
- the anchor value enlarges a small amount briefly.
- For an audio item a sound effect is given.

The style of a source anchor may depend on other properties of the link emanating from it, for example: whether the source context will disappear on traversing the link; whether the reader has already seen the destination of the link.

When a number of anchors are grouped together as the source of a link, the same style can be applied to them all. For example, when the user moves the cursor over any one of the anchors in the group all the anchors will, e.g., " pop-up".

The scenario in Fig. 2.13 illustrates a number of anchor styles.

Transition style

A *transition style* is the special effect that can be applied as a instance starts to play, fi nishes playing[SFHS91], or changes to another instance. For example, a video can " dissolve" to another video, or an audio fragment can " fade-out". Sound effects may also be part of the transition style. A transition style should not be limited to instances, but also applicable to compositions. For example, when a link is followed, the destination context can replace the source context using such effects as " wipe-down"or " zoom-in". The scenario in Fig2.13 illustrates the dissolve special effect.

Link style

This can be applied to links for representing them in network type diagrams. We do not discuss this further.

Dexter layers	Model features	Model requirements
Presentation Specifi cations	Styles	associate with anchors, instances, compositions, composite anchors and links

TABLE 2.10. Model requirements for styles

2.2.4.4 Activation state

When playing a presentation different parts of the document are made active, and then are deactivated. For example, playing a multimedia sequence activates a number of media items and after the specifi ed time, deactivates them. The activation state of the items follows from their temporal ordering within the presentation. Link navigation gives the reader the choice of activating other presentations. The activation state is then dependent on the selection of links carried out by the reader.

(i) Source anchor is displayed.
Anchor style is given in instance.

(ii) User selects anchor.

(iii) Source anchor highlights.
(Anchor border thickens and
text background changes colour.)
Anchor highlight style is given in link.

(iv) Source context dissolves into
destination context.
Transition style is given in link.

(v) Destination context displayed.
(Destination anchor not highlighted.)

Veel van de huizen en ...

Figure 2.13. Anchor and transition styles on following a link

Throughout the process of playing a presentation the player software requires a record of the activation state of the parts of the document. In particular, when starting up a hypermedia presentation from its document description the player has to make a decision as to which part or parts of the presentation should be displayed initially. In the example in Fig. 2.2 the initial screen on starting up the presentation is a single contents screen. If the document describes only a single multimedia presentation, then this can be started at the beginning and played through to the end. If the document describes more than one multimedia presentation then one or more of these can be activated upon start up. The requirement for the document model is that the initial activation state of the presentation is specifi ed within the document. When the document contains links as well as multiple multimedia presentations, the requirement for the document model is that the change in the activation state of the presentation is specifi ed within the document.

In addition to activation of individual multimedia presentations, a presentation can have two states—playing and paused. This may be for a complete pr esentation, or for a single continuous media item, for example a video. In other words, a continuous media item and a multimedia presentatoin each have their own intrinsic timeline which can be traversed, or not. The normal condition when playing a continuous media item or a presentation is the " playing"state, but it may also be in the " paused"state, i.e. its timeline is not being traversed. The presentation is still active, however. When a link is followed from an active presentation the source context can become inactive or can remain active and either continue in its playing or paused state or change to its paused or playing state. The destination of the link becomes active, and each presentation can start up in the playing or paused state. The activation state changes on following a link are summarised in Table 2.11.

Initial state of source	New state of source	Destination
Source playing	is replaced, continues playing, or pauses	becomes active (if not already) and plays or pauses
Source paused	is replaced, starts playing, or remains paused	becomes active (if not already) and plays or pauses

TABLE 2.11. Activation state change on following a link

The requirement for the document model is that the initial playing/paused state of the presentation is specifi ed within the document. When the document contains links as well as multiple multimedia presentations, the requirement for the document model is that the change in the playing/paused state of the presentation on following a link is specifi ed within the document.

Dexter layers	Model features	Model requirements
Presentation Specifications	Activation state	initial activation state, change in activation state initial play/pause state, change in play/pause state

TABLE 2.12. Model requirements for activation state

2.2.5 Runtime layer: Interaction

Interaction with an application covers a range from sitting passively while watching a multimedia presentation to inputting different forms of information for manipulating an on-line environment. In this section we state the boundaries of the expected reader interaction with a hypermedia presentation. We assume that the reader is supported by a runtime environment for playing the presentation. Interaction features do not influence the document model, but allow aspects of the presentation to be changed by the reader at runtime.

Aspects of a hypermedia document for which interaction support can be provided by a player are the temporal flow of the presentation, the spatial layout and the links. These are not requirements for the document model itself, but a consequence of it. We specify what should be available in a runtime environment. We briefly discuss more complex interaction with an underlying application.

2.2.5.1 Temporal control

The document corresponding to a multimedia presentation contains within it information on how fast the presentation should be displayed to the reader. Controls can, however, be given to the reader to vary the speed of presentation. For example, the presentation can be halted, played from any point during the presentation, speeded up, slowed down or played in reverse. The reader can be given control over the position of the presentation on its temporal axis and the speed and direction at which the presentation is played. Implementing fast forward and reverse control may require more than straightforward implementation however [HeKo95]. Halting the presentation can also be specified from within the document as in, for example, the "pause" command in Director and the "wait" command in Authorware.

In a runtime environment basic temporal control of a playing presentation needs to be given to the reader, for example halting the presentation and continuing it.

2.2.5.2 Spatial control

Spatial control is similar to temporal control, except that the spatial axis is comparatively shorter, i.e. it is not often the case that the reader is able to see only a small part of the media item and has to scroll through the rest. Nevertheless, control can be given to the reader in the form of zooming in and out of one or more spatial dimensions, and scrolling the presentation. A more interesting form

of spatial control is for three-dimensional objects where the orientation of the object is important to the reader, and a means for rotating the object in three dimensions should be supplied. Scrolling of text or large screen-based media items should be available to the reader.

2.2.5.3 Navigation control

A hypermedia document specifies links which can be expressed as navigation actions. The reader needs some means of selecting a link to follow, where the most common method is moving a mouse cursor over the desired object then selecting it. Another means is to use keyboard commands to move the cursor and make the selection. Functionality that should be supported beyond following links is recording where the reader has been and allowing the reader to return to previously seen material.

Links can be used for making choices which are similar to interactions with variables, but only in the case that the choices are finite. For example a game of noughts and crosses, otherwise known as tick-tack-toe, could be implemented using anchors and links by having a complete set of all possible stages of the game stored in the document. The reader plays the game by clicking on the desired empty square and so following the appropriate link. More complex processing than selecting from pre-authored information requires the creation of a specific application.

Application state control

A more general form of interaction can be specified between the reader and an underlying application. This allows the creation of flexible and elaborate interactions, but requires the support of input for different values of variables within an 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. Another example is the management game described in [HaRB95], where users can interact with a simulation of a financial market via an interface implemented as a hypermedia presentation. Here the values of the variables as entered by the reader are stored in the underlying application and the results of the simulation calculations are displayed as graphs within the presentation. The requirement for a hypermedia presentation environment is that communication should be possible between it and an external application.

2.2.6 Summary of hypermedia document model requirements

In this section we discussed a number of requirements for a hypermedia document model. A summary of these is given in Table 2.13. Note that the structure of the table reflects that of Table 2.1, where the runtime layer has been omitted. This is because any aspects of the runtime layer which influence the document model are recorded in the presentation specifications.

In the next section we show that, although hypertext and multimedia document models already exist, none is sufficiently powerful to meet all the requirements listed in this section.

Dexter layers	Model features	Model requirements
Within-component layer	Media items	temporal and spatial dimensions
Anchoring	Reference to part of media item	data-dependent specification of part of media item
Storage layer	Instance of media item	reference to (part of) media item, data format, duration, start time, extent, position, aspect ratio, orientation, Z-order, style (media item, anchor, transition), start points for links, semantic attributes
	Composition	temporal and spatial composition, space/time independent composition, grouping of anchors into composite anchors
	Linking	source and destination anchor, source and destination context, transition (duration and special effect) change in activation state change in playing/paused state
	Semantic attributes	associate with anchors, instances, compositions, composite anchors and links
Presentation Specifications	Temporal layout	time axis, start time, duration, scaling, Allen's relations, link transition duration
	Spatial layout	space axis, position, possibly changing w.r.t. time, extent, scaling, orientation, Z-ordering, link transition spatial relation
	Styles	associate with anchors, instances, compositions, composite anchors and links
	Activation state	initial activation state, change in activation state initial play/pause state, change in play/pause state

TABLE 2.13. Summary of requirements for a hypermedia document model

2.3 Existing hypertext and multimedia models

Systems for authoring and reading hypertext material have existed for a number of years. One of the earliest was Engelbart's Augment [Niel95]. Of the more well-documented systems are Notecards ([Hala88], [Niel95]) Intermedia ([HKRC92], [Niel95]) and Hyperties [Niel95]. HyperCard [Niel95] and Guide ([OWL90], [Niel95]) are commercial systems initially available in 1985. Unfortunately, each of these systems embodies its own, implicit, model of hypertext. One of the first formal models of hypertext is the Dexter hypertext reference model, [HaSc94], which was developed by a number of hypertext system designers to express the essence of a hypertext system. This model remains the most influential in the hypertext literature.

Multimedia authoring systems have also been available, although tend to have had more interest in the commercial world, for example Director and Authorware [Macr97]. Again each system has its own, implicit, model of multimedia. The CMIF multimedia model, [BuRL91], was explicitly developed, however, in order to capture the essence of a multimedia presentation. CMIF is one of the few explicit models of multimedia.

While the Dexter and CMIF models make important contributions towards creating a full hypermedia model, neither captures all the concepts necessary. A hypermedia model should be able to describe structured information incorporating multiple continuous and static media along with temporal and spatial relationships. It should not be restricted to dealing with small multimedia presentations, but should allow the re-use of smaller presentations in the creation of larger, more complex hypermedia presentations.

We first give brief descriptions of the Dexter and the CMIF models, and then compare them with the requirements from the previous section. We demonstrate where each model is insufficient as a hypermedia document model, and propose that a hypermedia model needs to be a superset of the two. We conclude with the requirements that are missing from both models that also need to be included in a hypermedia model.

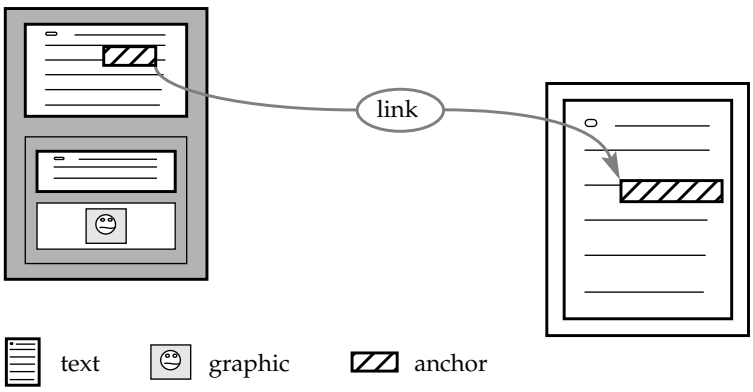
2.3.1 Dexter hypertext model

The Dexter hypertext reference model, [HaSc94], was developed as a reference model to rationalise and make explicit the concepts embedded in the then existing hypertext systems. The model defines a number of concepts present in a number of systems, but not all of which are present together in any one "Dexter compliant" system. At the time there was no existing implementation which encapsulated all of the Dexter model, although since then work has been carried out to implement the Dexter model as such, [GrTr94a]. The Dexter model is widely referenced, e.g. [Rada95] and [GrTr94b], and is the standard of comparison for hypertext systems.

The Dexter model divides a hypertext system into three layers: a *within-component layer*, where the details of the content and internal structure of the different media items are stored; the *storage layer*, where the hypertext structure is stored; and the *runtime layer*, where information used for presenting the hypertext is stored and user interaction is handled. The Dexter model describes the storage layer in detail, and it is this layer which is most relevant to a hypermedia model and of which we give a brief description here.

The Dexter model introduces atomic, composite and link components, and anchors. Atomic and composite components are related to each other via link components, where anchors specify the location of the ends of the links. This is shown schematically in Fig. 2.14.

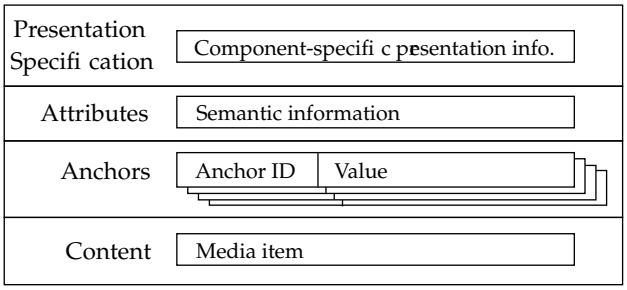
The components of the Dexter model are illustrated in detail in Fig. 2.15. A Dexter component has its own unique identifier, not shown in Fig. 2.15. A reference to a component can be made directly to its unique identifier or via a more general component specification, shown as "Component specifier" in Fig. 2.15. The latter requires a resolver function to "resolve" it to a unique identifier, for instance, to allow the addressing of a component by means of an SQL database query. A WWW URL, for example, can be a unique identifier or a component specification[7].



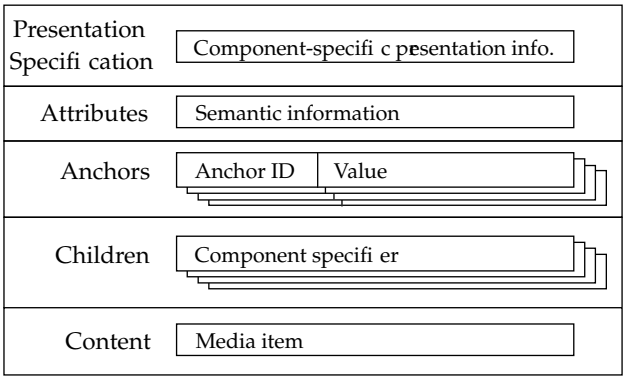
A composite component (left) is linked to an atomic component (right). (There is no representation of time in the figure.)

Figure 2.14. Dexter components schematic

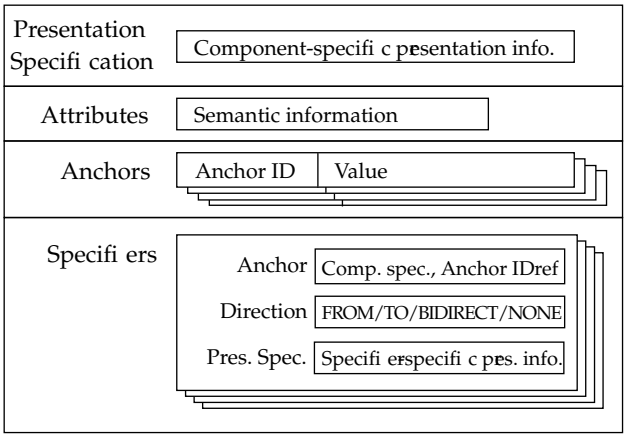
7. World Wide Web Uniform Resource Locator. A unique identifier is e.g. `<http://www.w3.org/Addressing/URL/>`, a component specification is e.g. `<http://www.altavista.digital.com/cgi-bin/query?pg=q&what=web&kl=XX&q=Hypermedia+Model>`.



(a) atomic component



(b) composite component



(c) link component

Figure 2.15. Dexter model

Atomic component
An *atomic component* contains 4 parts—presentation specification, attributes, a list of anchors and content, Fig. 2.15(a) (inspired by Fig. 4 of [HaSc94]).

- The *presentation specification* holds a description of how the component should be displayed by the system, the form of which is beyond the scope of the model.
- The *attributes* allow a semantic description of the component to be recorded. The form of this semantic information is beyond the scope of the model.
- An *anchor* is composed of an anchor identifier and a data-dependent anchor value. The anchor identifier is unique within a component and allows the anchor to be referred to from a link component. The anchor value specifies a part of the content of the atomic component. The anchor value is the only place in the model where the data type of the content is required.
- The *content* is a media item of a single data type.

Composite component
A *composite component* is the same as an atomic component with, in addition, a list of child components, Fig. 2.15(b)[8]. It is a collection of other components (atomic, composite or link) which can then be treated as a single component. This structuring of components is restricted to a directed, acyclic graph (by definition)[9]. The anchors of a composite component refer to the content of that component.

Link
A *link* is a connection among two or more components. Its structure is the same as an atomic component with a list of specifiers replacing the content, Fig. 2.15(c). A *specifier* defines an end-point of the link. It consists of an anchor, a direction and a presentation specification.

- The anchor refers to an anchor identifier in a component, shown as "IDref" in Fig. 2.15(c).
- The direction is one of FROM, TO, BIDIRECT or NONE. FROM and TO specify that the end-point referred to (via the anchor) is a source or destination of the link respectively. BIDIRECT allows the end-point to be both source and destination, and NONE allows neither.
- The presentation specification refers to the anchor style at an end-point of the link. A single link component can allow the expression of a range of link complexities, including a simple one-source, one-destination, uni-directional link (for example links in HTML), or a far more complex multi-source, multi-destination, bi-directional link.

8. A composite thus also includes content, although in the original Z specification, in the NIST publication of the model [HaSc94], a composite does not include content.
9. Although in the original Z specification, in the NIST publication of the model [HaSc94], the composition was restricted to a hierarchy.

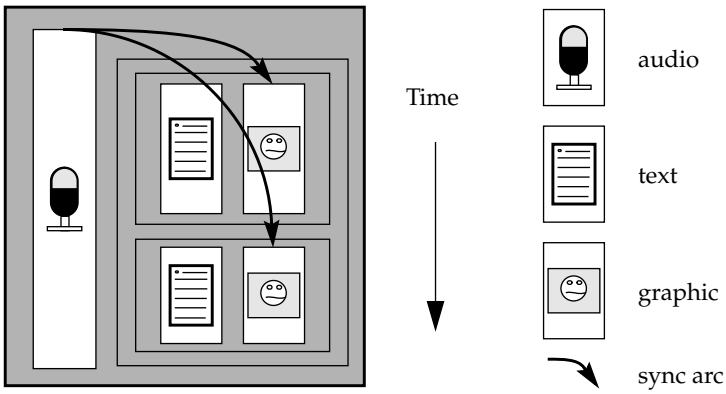
Note that atomic and composite components do not store a list of links which refer to them, and have only a list of their own anchors. It is left to the runtime layer to calculate which anchors are linked to within the scope of the hypertext environment. Anchors were introduced as a means of keeping links free from data-dependent issues, i.e. as a means of separating the storage and within-component layers. Note that since links can have anchors, links are able to link links. This has been implemented in [GrTr94a].

2.3.2 CMIF multimedia model

The CWI Multimedia Interchange Format (CMIF) model [BuRL91], describes a model for representing and manipulating multimedia documents. Multimedia document authoring systems have received scant treatment in the academic literature, until the introduction of the ACM conference on multimedia. The CMIF model is one of the first, and still one of the few, multimedia document models.

The CMIF model captures the time-based organisation of information, and allows the construction of larger documents out of smaller documents (hierarchical structuring). The resulting document is a continuous presentation.

Important parts of the model, for this discussion, are that documents are compositions of atomic and composite nodes, an atomic node is presented via a channel, and synchronization arcs can specify timing constraints between two atomic nodes, Fig. 2.16.



A parallel composite component has two children: an audio atomic component and a sequential composite. The sequential composite has two parallel composite children each containing a text and a graphic atomic component. Both graphic atomic components have their starting times constrained by synchronization arcs.

Figure 2.16. CMIF components schematic

Atomic node
An *atomic node*, Fig. 2.17(a), consists of a presentation specification, attributes and content.

- The presentation specification is in essence the same as in the Dexter model, but is split into two parts. The first part, the channel, specifies via which channel the node will be played, and the second, the *event descriptor*, specifies the presentation of one instance of the content. The latter can supplement or override the channel information.
- The *data descriptor* is a set of attributes describing the semantics of the data block (the same as the attributes in the Dexter model).
- The *content* is the same as in Dexter, specified as a media item. This is the smallest unit that can be mapped onto a channel for presentation.

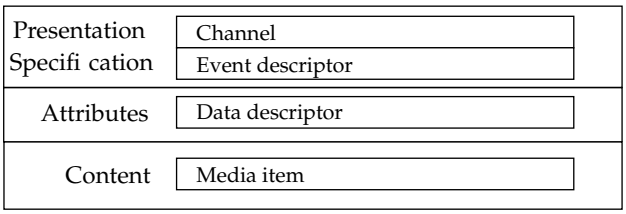
Channel
A channel is an abstract output device for playing events. This may be, for example, a window on the screen, or audio output. The channel includes default presentation information, for example font and style for a text channel, or volume for an audio channel. The only means of playing content is via a channel. The number of channels within a document is not restricted. When a document is played the channels are mapped onto physical output devices.
The channel is a means for specifying multimedia styles, in particular media item styles.

Composite node
A *composite node*, Fig. 2.17(b), is a (strictly) hierarchical composition of atomic and composite nodes. The *type* specifies whether the composition is SEQUENTIAL or PARALLEL. In a SEQUENTIAL composite the child nodes are played one after the other, in a PARALLEL composite they start at the same time.

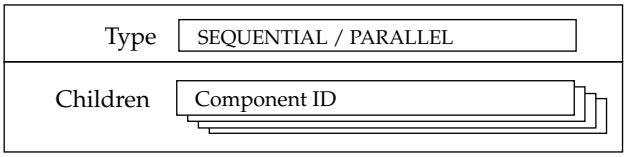
Synchronization arc
A synchronization arc, Fig. 2.17(c), specifies system architecture-independent timing constraints between two atomic nodes. The *source* is the atomic node at the beginning of the arc and the *destination* that at the end. The *scheduling interval* gives the delay that should occur between the presentation of the two ends of the arc. This consists of the mid-point scheduling time, that is the desired delay, and allows variations to account for possible system delays when playing the presentation. The *synchronization type* specifies how the interval should be interpreted (START, END, OFFSET) and how important it is that the synchronization condition should be met.

2.3.3 Comparing Dexter and CMIF with the model requirements

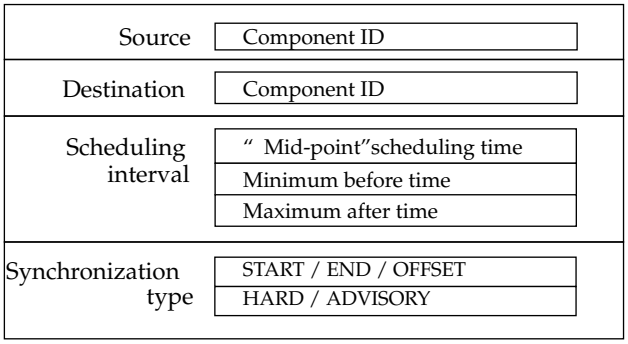
Having given descriptions of the Dexter and CMIF models we compare them with the requirements derived in Section 2.2. This is summarised in Table 2.14.



(a) CMIF atomic node



(b) CMIF composite node



(c) synchronization arc

Figure 2.17. CMIF model

2.3.3.1 Limitations of Dexter

The Dexter model allows the composition of hierarchical structures and the specifi cation of links among components. Defi ciencies of the model for expressing structured hypermedia information incorporating multiple continuous and static media along with temporal and spatial relationships are the following.

- The main drawback of the model is that while it accommodates the inclusion of continuous media items at the within-component layer it does not include time at the structuring level of documents—the storage layer . This

Dexter layers	Document model features	Requirements	Dexter	CMIF
Within-component layer	Media items	temporal and spatial dimensions	no	yes
Anchoring	Reference to part of media item	data-dependent specifi cation of part of media item	yes	no
Storage layer	Instance of media item	reference to media item, data format,	yes	yes
		duration and start time,	no	yes
		extent, position, aspect ratio, orientation, Z-order	no	partly
		style (media item, anchor, transition),	no	partly
		start points for links,	yes	no
		semantic attributes	yes	yes
	Composition	space/time dependent composition,	no	yes
		space/time independent composition,	yes	no
		grouping of anchors into composite anchors	no	no
	Linking	source and destination anchor,	yes	no
		source and destination context,	no	no
		transition (duration and special effect)	no	no
	Semantic attributes	associate with anchors, instances and compositions	yes	yes
Presentation Specifi cations	Temporal layout	time axis,	no	yes
		start time,	no	yes
		duration,	no	yes
		scaling,	no	no
		Allen's relations,	no	yes
		link transition duration	no	no
	Spatial layout	space axis,	no	yes
		position,	no	yes
		extent,	no	yes
		spatial relations,	no	no
		scaling,	no	yes
		orientation,	no	no
		Z-ordering,	no	no
		link transition spatial relation	no	no
	Styles: media item, anchor, transition	associate with anchors, instances, composites and links	partly	partly
	Activation state	initial activation state	no	yes

TABLE 2.14. Comparison of Dexter and CMIF with requirements

is the difference between temporal and atemporal composition discussed in Section 2.2.3.2.

- A second major drawback is that a link end is specifi ed only by an anchor and includes no larger information context for the link end-point, as discussed in Section 2.2.3.4. Nor does it include a specifi cation of what happens to the source context of the link when a link is followed.[10]
- Composite anchors are not explicitly included in the model.[11]

- A Dexter composite includes content, Fig. 2.15(b). This means that the presentation specifi cations (and other component attributes) apply ambiguously to either the data in the current component or all the descendants of the component.
- Dexter anchors have no explicit semantic information associated with them, i.e. no equivalent of a component's attributes.

2.3.3.2 Limitations of CMIF

The CMIF model allows the hierarchical composition of static and continuous media into time-dependent presentations. Defi ciencies of the model for expressing structured hypermedia information incorporating multiple continuous and static media along with temporal and spatial relationships are the following.

- The major drawback of the model is that it does not include links and anchors.
- The model does not incorporate atemporal composition.
- Anchor attributes are not included in the model.

2.3.4 Requirements missing from both Dexter and CMIF

A number of requirements for a hypermedia document model are in neither the Dexter nor the CMIF models. The following are a consequence of the combination of multiple synchronized media items with links among these synchronized groups.

- Both temporal and atemporal composition are required within a single model. Atemporal composition is required for providing choices which can be linked to. Temporal composition is required for synchronizing multiple items.
- Activation state information is needed, both for the play/pause state of any single temporal composition and for the specifi cation of how the activation state of the presentation changes on following a link.
- Context for linkends is needed for specifying the scope of the ends of a link.
- Transition information for a link is required, since on traversing a link the presentation as perceived by a reader should remain a continuously playing multimedia presentation.

Other requirements missing from both models are the following.

---

10. A leave/replace attribute is also necessary in the Dexter model, but this was omitted. Whether the source component of the link was left on the screen or was replaced was often implicit in the system implementation, for example Notecards [Hala88] and Hypercard [Niel95]. It could also be tied to the link type, for example in Guide [OWL90], [Niel95] a note button left the source context and displayed the destination context in a separate window, whereas the inquiry button cleared the source context.
11. Dexter *could* allow this by using the value fi eld of an anchor to specify a (list of) ComponentID/AnchorID pair(s), but this was not the original intention of the model.

- Part of a media item should be specifi able as the content of an atomic component to allow re-use of data, in particular for large media types such as video.
- Attributes for anchors are needed because although a media item is an atomic entity in a computer system (the smallest amount of data that can be retrieved from a data store) it is not necessarily a single real-world object. A single media item may portray a number of real-world objects, so that attributes on anchors are necessary for labelling them individually.
- Composition of anchors is needed for grouping together semantically equivalent parts of different media items.

2.4 Conclusion

We began this chapter by introducing an example of a hypermedia presentation. In order to create a hypermedia document model capable of expressing this type of hypermedia presentation, we need to satisfy the requirements stated in Section 2.2. Our summarised list of requirements for a full hypermedia document model is the following.

- The specifi cation of a media item needs to include information about the dimensionality of the media item, in particular whether it has intrinsic spatial or temporal extents.
- Data dependencies of the media item should be encapsulated in an instance, along with other properties describing the media item.
- Composition of components should include space/time dependent composition for the creation of continuous multimedia presentations composed of synchronized subcomponents, as well as space/time independent composition for the grouping of separate presentations among which a reader can navigate.
- Anchors are required for the localisation of data-dependent specifi cations of parts of media items.
- Links are necessary for the specifi cation of relationships among multimedia presentations. They differ from hypertext links in two ways. Firstly, given that multiple media items are collected together in a presentation, a link context is necessary to specify the scope of the linkend. Secondly, because of the temporal nature of the presentation, a pause/play attribute is necessary for the case that the source context is not replaced and for the destination context.
- Temporal layout is required for synchronizing multiple components into one continuous presentation.
- Spatial layout is required for determining where the media items are to be displayed on the screen.

- Styles for media items, anchors and transitions are required for specifying different presentation aspects of parts of media items, instances, compositions and links.
- Attributes are not required for the specification of a single presentation, but are required for more content-based creation of presentations and retrieval of already created presentations.

While Dexter and CMIF are adequate models for describing hypertext and multimedia, respectively, we have shown that both lack aspects for a complete hypermedia model. Some superset of these two models is thus needed in order to be able to describe synchronized multiple media including links. Of the requirements for a hypermedia document model, those that are not already part of the Dexter or the CMIF model are:

- context for linkends, required because of the multiple components collected together in the source and destination contexts;
- transition information for a link, since on traversing a link the presentation should remain a continuous presentation as perceived by the reader;
- attributes for anchors, since these portray basic real-world objects, in contrast with a media item which is a basic system object.

While these are our requirements for a complete hypermedia document model, a model that satisfies them also has to remain sufficiently simple to be implementable, while remaining sufficiently expressive for the majority of documents. A model satisfying these requirements has to find an appropriate compromise between these two contradictory requirements. Our own solution to defining a particular model is given in the following chapter.

3 The Amsterdam Hypermedia Model

We define the Amsterdam Hypermedia Model (AHM) based on the requirements in Chapter 2. The model is first defined per element of the model followed by a discussion of the interdependencies of the presentation specifications—temporal layout, spatial layout, style information and activation state. The model is shown to satisfy the requirements stated in Chapter 2. The document models implicit in a selection of hypertext, multimedia and hypermedia systems are described in terms of the AHM, thus demonstrating that the model is sufficiently rich for expressing a range of presentations.
This chapter is based on work presented in [HaBR93a], [HaBR93b], [HaBR94], [HaBu97].

3.1 Introduction

This chapter defines a hypermedia document model, the Amsterdam hypermedia model (AHM), demonstrates that the model satisfies the requirements given in Chapter 2, and shows that the requirements are sufficient for describing a range of hypermedia presentations, see also Table 3.1.

The model is first defined per element after which the presentation specifications common throughout the model are discussed. This provides insight into how the different elements of the model are interdependent. The presentation aspects discussed are temporal layout, spatial layout, style information and activation state. These aspects are of fundamental importance in a hypermedia document description where presentation aspects have semantic as well as aesthetic consequences.

In order to demonstrate that the AHM is able to express the presentations created by a wide range of authoring systems, we select a number of representative hypertext, multimedia and hypermedia systems and describe their implicit document models in terms of the AHM. In order to demonstrate that the model is sufficiently simple to be implementable we supply a description of the parts of the model which have been implemented in the CMIFed environment.

The structure of this chapter is as follows. Section 3.2 gives the definition of the AHM. Section 3.3 discusses the runtime interdependencies of the model elements. Section 3.4 gives a summary of the model, shows that the model meets

the requirements from Chapter 2, and discusses the model as a whole. Section 3.5 describes a number of hypertext, multimedia and hypermedia systems and describes their implicit document models in terms of the AHM. Appendix 1 contains a specification of the parts of the model which have been implemented in CMIFed.

3.2 The Amsterdam hypermedia model

This section defines the Amsterdam hypermedia model. It is ordered in terms of the elements of the model: media item, channel, atomic component, composite components, link component.

We begin with an overview of the requirements from Chapter 2 and their correspondence to the elements in the model, shown in Table 3.1. Each document element description states the definition of the element, states the requirements for that element in a document conforming to the model, presents the requirements in tabular form, and discusses various aspects of that element. The requirements for a document specified in terms of the model are collected together in Table 3.6.

3.2.1 Media item

A *media item* is an amount of data that can be retrieved as a single object from a store of data objects or is generated as the output from an external process. The form of a media item is outside the scope of the AHM. For a media item to be included in a document the requirements laid upon it are that the temporal and spatial dimensionality are known and that the corresponding duration and extent are also calculable. The duration may be specified as indefinite.

3.2.2 Channel

A *channel* defines a spatial position and extent and collects together a number of presentation and semantic attributes that are applicable to a particular media type. A channel consists of an identifier, a presentation specification, attributes and a media type, Fig. 3.1.

The *identifier* is a globally unique identifier

The *presentation specification* stores a channel reference[1], spatial information for visual media types, and style information.

- The channel reference is a reference to another channel or a system-defined window.
- The spatial information specifies the position and extent of the channel. The position and extent are specified with respect to another channel, given by a channel reference, or a window.
- Style information includes media item style, anchor style and transition special effect.

AHM elements / Requirements		**Channel** (section 3.2.2)			**Atomic component** (section 3.2.3)						**Composite components** (section 3.2.4)						**Link component** (section 3.2.5)					
							Anchors						**Anchors**						**Anchors**			
		Pres. spec.	Attribtues	Media type	Pres. spec.	Attribtues	Pres. spec.	Attributes	Value	Content	Pres. spec.	Atributes	Pres. spec	Attributes	Value	Children	Pres. spec.	Attrbutes	Pres. spec	Attributes	Value	Specifi ers
Within component layer	Media item			*						*												
Anchoring	Part of media item								*	*												
Storage layer	Properties of instance				*	*	*	*														
	Composition: temporal										*					*						
	atemporal										*				*	*						*
	spatial	*																				
	Linking											*										*
	Semantic attr.		*			*		*				*		*				*		*		
Presentation Specifi cations	Temporal layout				*		*				*						*					
	Spatial layout	*			*												*					
	Style:																					
	media item	*			*						*											*
	anchor	*			*		*				*		*									*
	transition	*			*						*						*					*
	link																*					
	Activation state				*						*											*

TABLE 3.1. Hypermedia document model requirements and the elements of AHM

1. Throughout this chapter we will adhere to the standard terminology as it is evolving in the literature on hypermedia models. There are several exceptions, however, which we note here.
We use the term *component reference* to replace the Dexter term component specifi cation. This is to prevent overloading of the word specifi cation. We introduce *media item reference*, *anchor reference* and *channel reference* in addition to the component reference. These have the equivalent meaning of Dexter indirect addressing, allowing a search which returns a globally unique identity (UID) for the object. The corresponding resolver functions are implicit in the model.
Relationship terminology, such as child, parent, ancestor and descendant, is in terms of the document structure, and is not an object-oriented class structure.

The channel presentation specification may include default properties for the spatial and style information applicable to the media type associated with the channel, such as a scale factor, Z-order, orientation, background colour etc.

The *attributes* allow semantic information to be associated with the channel, for example the natural language used (for text or audio channels). A description of the attributes themselves falls outside the scope of the AHM.

The *media type* is the specification of one or more data formats that can be played on the channel.

Document requirements

The identifier and media type are required for all channels. The channel reference, position and extent are required for channels with a visual media type. The style and attributes are optional. The channel reference, position and extent are meaningless for non-visual media types. The document requirements are summarised in Table 3.2.

Model elements			Required	Optional
Channel	Presentation Specification	Channel ref. Position & extent Style	*[a] *[a]	 *
	Attributes			*
	Media type		*	

TABLE 3.2. Channel and required/optional document specifications

a. For a visual media type.

Discussion

The AHM channel is based on the CMIF channel, where the components of the channel are more explicitly specified. A spatial hierarchy is defined top down from a system defined window The channel referencing structure is a strict hierarchy.

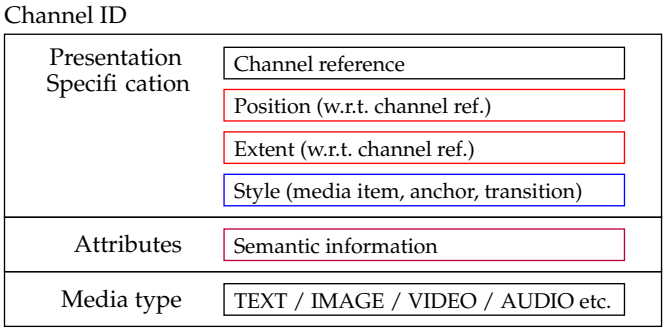


Figure 3.1. AHM channel

Associating semantic attributes with a channel allows complete streams of information to be selected, independently of the composition structure of the document.

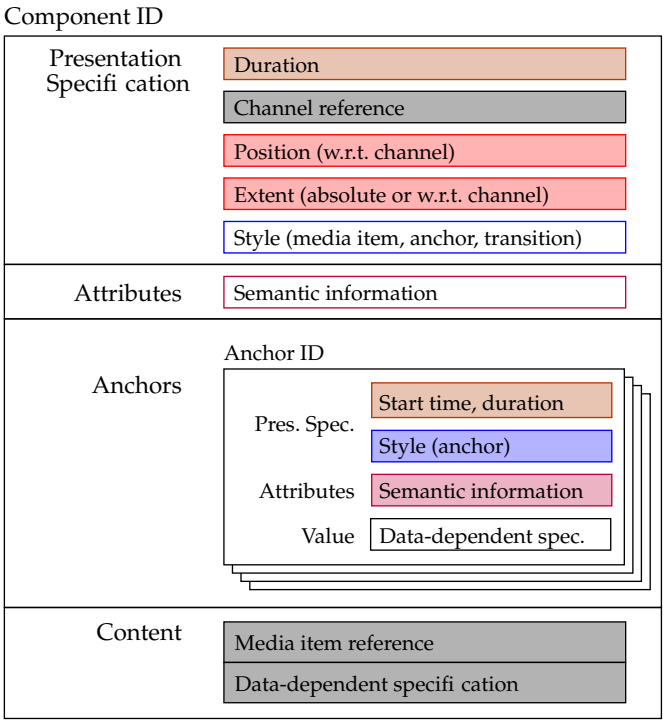
We postpone the discussion of the style aspects of channels until Section 3.3.2.

3.2.3 Atomic Component

An *atomic component* specifies information pertinent to a media item, including the data needed for displaying it, Fig. 3.2. It consists of an identifier, a presentation specification, attributes, anchors and content.

The *identifier* is a globally unique identifier

Semantic *attributes* can be associated with the component. They enable the creation of knowledge structures and information retrieval of the components. A description of the attributes themselves falls outside the scope of the AHM.



Changes from Dexter are shaded.

Figure 3.2. AHM atomic component

Presentation specification
The *presentation specification* consists of temporal, spatial and style information.

- Temporal information for an atomic component is the duration of the display of the content. The duration can be derived from the content and associated data format of a continuous media type, or can be explicitly specified for any media type. If the derived and specified durations are not the same (for a continuous media type) then the duration that is propagated throughout the model is the specified duration.
- Spatial layout is the position and extent of the display of the content, its orientation, its aspect ratio and its Z-order. The extent of the component can be derived from the data format of the content, can be specified relative to the channel reference or can be assigned an absolute value. The position is specified relative to the channel reference. Other layout information can be assigned through a channel reference or can be assigned to the component itself. The position, or other layout information, may be a function of time.
- Style information applicable to an atomic component is media item style, anchor style and transition. The style information can be assigned through the channel reference or can be assigned to the component itself. The style information can be media item, anchor and transition special effect.
  Transition information can be associated with the beginning and end of the display of the associated content. This requires the specification of a special effect along with a duration. This duration is part of the transition and is independent of the duration or start time of the component.

Anchors
Anchors form a media-independent interface between links and the components. The anchor structure for an atomic component is made up of an anchor identifier, a presentation specification, semantic attributes and a value, Fig.3.2.

- The anchor *identifier* is unique within the component and is used for referring to the anchor from other components by specifying its containing component identifier along with the anchor identifier
- The *presentation specification* specifies temporal information and anchor style applicable to the anchor. The former allows the association of a start time and a duration to the anchor. This is applicable only to a non-continuous media type, since otherwise the anchor value determines this implicitly.
- Semantic *attributes* can be assigned to an anchor. Attributes associate concepts directly with the data representation of a real-world object, which allows the creation of knowledge structures and information retrieval. A description of the attributes themselves falls outside the scope of the AHM.
- The anchor *value* specifies a part of the content of an atomic component using a data-dependent specification. If the content is not the complete media item associated with the component then the anchor value is

restricted to be within the bounds of the data-dependent specification of the content.

Content

The *content* specifies the data for the atomic component and consists of a media item reference, which is operating system dependent, and a data-dependent specification of part of the media item, which is data type dependent.

Document requirements

The parts of an atomic component that are required are the channel reference, the duration of the presentation specification and the content. The duration may be derived from the content. The other presentation specifications, attributes and anchors are optional. The content requires a media item reference and the data dependent specification is optional. The content need not be specified if a knowledge structure only is being created, but in this case the attributes are required. An anchor specified within an atomic component requires an anchor identifier and a value. Anchor attributes and presentation specifications are optional. The anchor value need not be specified if a knowledge structure only is being created, but the attributes would be required. The document requirements are summarised in Table 3.3.

Model elements			Required	Optional
Atomic Component	Presentation Specification	Duration	*[a]	
		Channel ref.	*	
		Position		*[b]
		Extent		*[b]
		Style		*
	Attributes			*
	Anchors			*
		Anchor ID	*	
		Pres. Spec.		*
		Attributes		*
		Value	*[c]	
	Content	Media item ref.	*[d]	
		Data-dep. spec.		*

TABLE 3.3. Atomic component and required/optional document specifications

a. The duration may be indefinite or unpredictable.
b. For a visual media type.
c. Each anchor value specification is restricted to being within the content, i.e. within the data-dependent specification of the media item reference.
d. The intrinsic duration and spatial extent are also known.

Discussion
The AHM atomic component is based on the Dexter atomic component with more detailed specification of the content and of the presentation specification.

Semantic attributes serve the same role as in Dexter. Our intention is to use these for providing cataloguing of the atomic component so that it can be found later in an information retrieval process and for creating semantic knowledge structures. An approach to this is detailed in [WBHT97].

The presentation specification for a component in Dexter is not specified in any detail. We have chosen to explicitly separate out temporal and spatial layout information from other possible presentation specifications including style information. Recording the temporal and spatial layout information is required so that components can be combined together in multimedia presentations, where the technical (as opposed to the artistic) problem is to pack items with temporal and spatial extents into a three dimensional space—or four dimensional, if the presentation consists of three dimensional objects, such as those in virtual reality applications.

The duration in the presentation specification of an atomic component is one of the document requirements. For continuous media, however, the duration can be deduced from the content and its data format, making a specified duration redundant. Without a specified duration, the rest of the environment would need access to the content data and a means of interpreting it in order to calculate the duration, thus we require its specification in the model.

The AHM atomic anchor is based on the Dexter anchor, with the addition of semantic attributes and presentation specification. Neither AHM nor Dexter specifies the form of the identifier or the value. The presentation specification for an atomic anchor allows a start time and a duration to be associated with an anchor referring to content of a non-continuous media type. For example, a music score could have each note as an anchor value and have each highlighted from a particular time for a particular duration. This would be recorded as the start time and duration in the presentation specification for the anchor value referring to a note. The anchor attributes allow knowledge structures to be created and searches to be carried out at a finer level of detail than complete atomic components, [BuKW94].

Allowing anchors to have a duration leads to a number of uses. One of these is to allow different links from the same non-continuous media item whose source anchors have the same anchor value but different start times. A second is to allow the consecutive highlighting of different anchor markers[2], as described in the music example above. A third is to enable synchronization between two streams so that the start times and durations of anchors in a non-continuous medium, e.g. written text, are dependent on the timing of corresponding

2. "Link marker" in Dexter.

anchors in a continuous medium, e.g. a spoken commentary. This would require two synchronization arcs from each audio anchor to each text anchor.

An anchor value is assumed to have a data-dependent definition. This is useful for the case that a data type can have anchor values specified within it, such as HTML. An alternative is to allow explicit spatial and temporal extents to be specified, so that a part of, e.g. an image, can be given in terms of its location on screen rather than in terms of the image data. The disadvantage of this is that if the source data is edited there is no record of where the anchor value should still be.

An assumption about an anchor value that should not be made is that it is a single temporal interval or a connected area. A single atomic anchor in an image could be, for example, all the leaves of a tree. In a video, an anchor may be all occurrences of a bouncing ball throughout the video. Neither the spatial nor the temporal extents need be contiguous.

In AHM, the content is not directly included within the atomic component, in contrast with Dexter which assumes that the data associated with the component is contained within the component. Including the content directly is seen as a disadvantage in [GrTr94a], and in the CMIF model it is explicitly stated that data can be contained within the node or stored as a reference from the node. The advantage of allowing references to the data become more apparent with larger media types, such as video, where a database can be tuned to the idiosyncrasies of the data type. Referencing the data also allows media items, or parts of items, to be re-used in other components with different semantic or presentation characteristics.

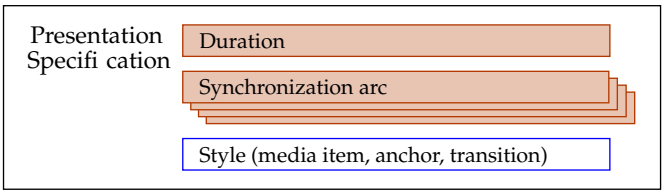
In AHM, the content is referenced using a media item reference rather than an identifier This allows media items to be dynamically selected for inclusion in a presentation, e.g. for including the most recent photo of a head of state or for allowing alternative data formats to make appropriate use of available network bandwidth.

3.2.4 Composite components

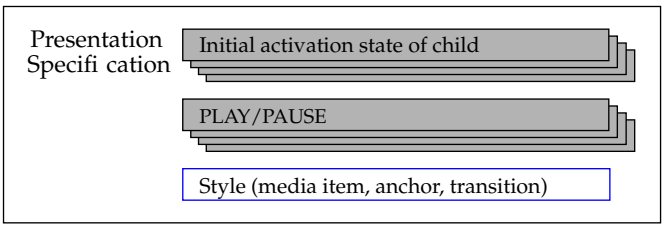
An AHM *composite component* is a single element referring to a collection of atomic, composite and/or link components. The composition types in AHM are temporal and atemporal. Temporal composition is a grouping of components which are temporally related to one another. Atemporal composition is a grouping of components with no associated temporal relations.

As with the atomic component, an AHM composite component has an identifier, a presentation specification, attributes and anchors, but instead of content a list of children, Fig. 3.3. We specify the presentation specification of temporal and atemporal composites separately.

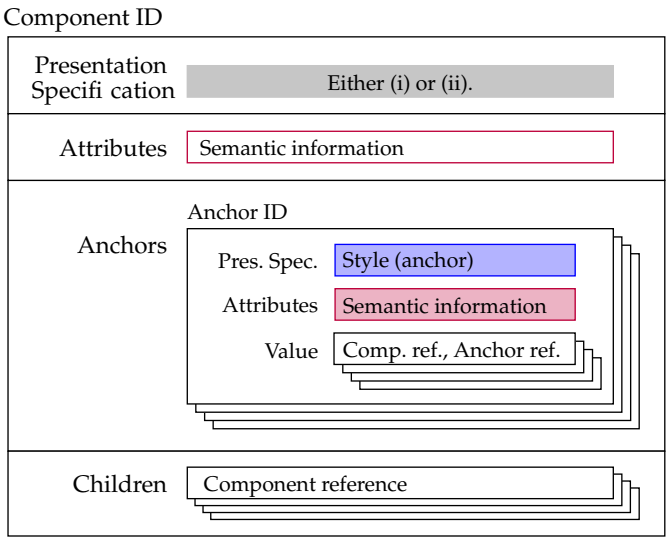
The *identifier* is a globally unique identifier



(i) Temporal presentation specifi cation



(ii) Atemporal presentation specifi cation



Composite component

Changes from Dexter are greyed.

Figure 3.3. AHM composite components

The *attributes* allow the attachment of semantic information to the composite as a whole, for the creation of knowledge structures and for retrieval purposes. A description of the attributes themselves falls outside the scope of the AHM.

Temporal composite presentation specification
The *presentation specification* for a temporal composite consists of temporal and style information, Fig. 3.3(i). The temporal information consists of synchronization arcs and a duration.

- A *synchronization arc* specifies the timing constraint between two parts of a presentation and in doing so establishes a single time axis shared by the ends of the arc. It consists of a source and destination, a scheduling interval and a synchronization type, Fig. 3.4. The source and destination specify anchors which are the end-points of the temporal relation. Each is a component reference/anchor reference pair plus a START or END attribute. The anchor may be a point in time, but will more likely be an interval, although not necessarily contiguous. The START/END attribute specifies whether the scheduling interval starts from the beginning or the end of the source anchor reference and extends to the beginning or the end of the destination anchor reference.
  The scheduling interval specifies the temporal relation between the source and destination of the arc. The start time of the destination is relative to the source.
  The synchronization type specifies tolerance and precision properties.
  The source and destination component references are restricted to referring to a component which is a descendant of the temporal composite or the temporal composite itself.
  The children of a temporal composite with associated content require to be located along the same time-axis. This is achieved by the association of synchronization arcs. The intrinsic duration of the composite is the result of combining the durations of the children along with all the specified synchronization arcs. The duration may be indefinite.
  The synchronization arc does not have an identifier, since it is meaningful only within the temporal composite in which it is specified.
- The *duration* allows a scaling factor to be applied to the duration calculated from the duration of the children of the composite and the synchronization arcs.
- *Style* information can be specified to apply to all the descendant atomic components of the composite. It can contain media item style, anchor style and transition special effect. It may also include link style if the composite includes links.

Atemporal composite presentation specification
The *presentation specification* of an atemporal composite consists of an initial activation state and style information, Fig. 3.3(ii).

- The *initial activation state* specifies whether each child is played or not when the composite is activated at runtime. The *play/pause state* is the initial state of the child when it is made active. Each child of the composite requires an initial activation and play/pause state.
- *Style* information can be specified to apply to all the descendant atomic components of the composite. It can contain media item style, anchor style, transition special effect and link style.

Anchors
The *anchor* for a composite component has the same structure as the anchor for an atomic component: identifier, presentation specification, attributes and value.

- The *identifier* and *attributes* are the same as for an atomic component anchor.
- The *presentation specification* specifies an anchor style for the descendants of the anchor.
- The *anchor value* is a list of component reference/anchor reference pairs where the component reference refers to a component which is a descendant of the composite component. The component reference can refer to an atomic or composite component. The anchor reference may be omitted, with the interpretation that the complete component plays the role of the anchor. This removes the requirement for introducing a special anchor value for referring to a complete component. The structure of the composite anchor is a hierarchy, since the composition of components is a directed acyclic graph and because of the descendant restriction on the component reference.

Children
The *children* are the components grouped together to form the composite and are given by a list of component references.

Document requirements
A temporal composite requires at least one child with associated content (directly or indirectly). The structure composed of the children and the synchronization arcs must specify a single temporal extent, in other words the compo-

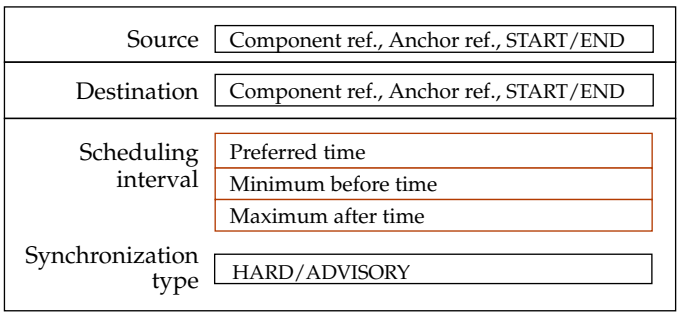


Figure 3.4. AHM synchronization arc.

nents and synchronization arcs must form a connected graph. An explicit duration, extra synchronization arcs, styles, attributes and anchors are optional.Each synchronization arc requires a source and destination component reference/anchor reference and START or END attribute, where both components are descendants of the temporal component from which they are referred to, and a preferred scheduling interval. The source and destination cannot use the same component reference. The synchronization type is optional.

An atemporal composite requires at least one child. Each child is required to have an associated initial activation state and play/pause state. Synchronization arcs cannot be specifi ed among descendants of an atemporal composite since the descendants are temporally independent. Styles, attributes and anchors are optional. The initial activation and play/pause states may be omitted if a knowledge structure only is being created, but in this case the attributes are required.

An anchor specifi ed within a composite component requires an identifi er and at least one component reference/anchor reference pair. The anchor reference may be omitted from the component reference/anchor reference pair. Style and attributes are optional. The anchor value may be omitted if a knowledge structure only is being created, but in this case the anchor attributes are required.

The document requirements are summarised in Table 3.4.

Model elements			**Required**	**Optional**
Temporal Composite Component	Presentation Specifi cation	Duration		*
		Sync. arcs	*	
		Style		*
	Attributes			*
	Anchors			*
		Anchor ID	*	
		Pres. Spec.		*
		Attributes		*
		List of anchors	*	
	Children	Comp. ref.	*[a]	
Atemporal Composite Component	Presentation Specifi cation	Initial activ. state	*[b]	
		Play/pause	*[f]	
		Style		*
	Attributes		As for Temporal Composite	
	Anchors		As for Temporal Composite	
	Children		As for Temporal Composite	

TABLE 3.4. Composite components and required/optional document specifi cations

a. At least one is required.
b. One per child is required.

Discussion
The AHM composite component is based on the Dexter composite component. The AHM, however, references its child components rather than including them, which is the approach taken in Dexter. The AHM gives the advantage of being able to include the same (fully-fledged[3]) component in multiple composite components. The AHM differs slightly from the work in [GrTr94a], where the authors allow both referenced and included components as children of the composite, since this allows any component referenced by a composite also to be referenced by any other composite.

In contrast to Dexter[4], we exclude content from being associated with the composite component, since this introduces ambiguity about whether the other properties of the component (in particular the attributes and presentation specification) apply to the content or to all the descendants of the component. We thus require an extra atomic component to be created around the composite's "content" and have it included in the composition with no special status.

The structure composed of the children and the synchronization arcs must specify a single temporal extent. This requires the specification of a synchronization arc for all but one of the children of the composite.

More than one synchronization arc per child may not be meaningful. Two cases can be distinguished:

(a) the destination component has a finite duration, so that the start and end time of the destination is fully specified with respect to the source;

(b) the destination component has an indefinite duration, so that as well as specifying the start time of the duration with respect to the source, the end time of the destination can also be given in terms of the source. For example, a text item may be specified to start 3 seconds before the end of a video and continue until 2 seconds after the end.

A synchronization arc can be regarded as a specialised link type, since it has two atomic or composite components as end points and other pieces of scheduling information that could be collected together in the link's attributes. We, however, prefer to keep it separate in the same way that Dexter separates attributes of a component from the presentation information. In other words, a link defines a semantic relation among components, whereas a synchronization arc defines a presentation relation (specifically timing). The synchronization arc specifies a constraint between two components sharing the same time axis. A link can connect components which are otherwise unrelated in terms of space and time.

The AHM composite anchor is a new construct. It has been noted that anchoring in composites in Dexter is underspecified[LeSc94], since it does not provide

3. Dexter's composite is a composition of base components plus one set of presentation specification, attributes and anchors.
4. Although the Z specification of the Dexter model does not express this.

semantics for attaching links to embedded atomic components in composite components. Similar problems are also identified in [GrTr94a], where the authors ask specifically whether an anchor in the parent composite can be tied to an anchor in one of its components. By introducing composite anchors we are able not only to re-use anchors in a single atomic component, but also group anchors together, associate new semantics and presentation specifications to the grouping, and attach a link to the group. Note that the children of an anchor of a temporal composite may not be active at the same time, e.g. the same object appearing several times during a sequence of video clips could belong to the same composite anchor. An atemporal composite may also have composite anchors, in which case the activation state of the presentation determines which of the children of the anchor are active.

Dexter uses an identifier reference for referring to an existing anchor. Since we have extended anchors to include attributes, it becomes useful to refer to an anchor on the basis of its attributes as well as via its identifier. This requires the use of an anchor reference rather than an anchor identifier. We thus use the anchor reference in a composite anchor.

An anchor style is often medium dependent (for example underlining a word in text). For a composite anchor, styles are needed which are not necessarily medium independent, but should be functionally equivalent for different media types. For example, for the style "emphasize a destination anchor" text is underlined, an image has a box and sound contains a beep.

An atomic component has of itself no explicit start time. This is captured in the temporal composition structure via the synchronization arcs. For an atemporal composite the children are not related in any temporal way, so that the start time of any child is determined at runtime.

An atemporal composite has no duration. Each of its children may have its own duration if it is a temporal composite or an atomic component.

3.2.5 Link component

A *link component* specifies a relationship among components (atomic, composite or link), Fig. 3.5. It consists of an identifier, a presentation specification, attributes, anchors and a list of specifiers.

The *identifier* is a globally unique identifier.

Semantic *attributes* can be associated with the component, in particular for describing the relationship the link represents.

Presentation specification

The *presentation specification* consists of a duration, a relative position and style information.

- The *duration* is the duration of the transition of the source context to the destination context of the link (the definition of source and destination context is given below). It is specified using a synchronization arc, where the source

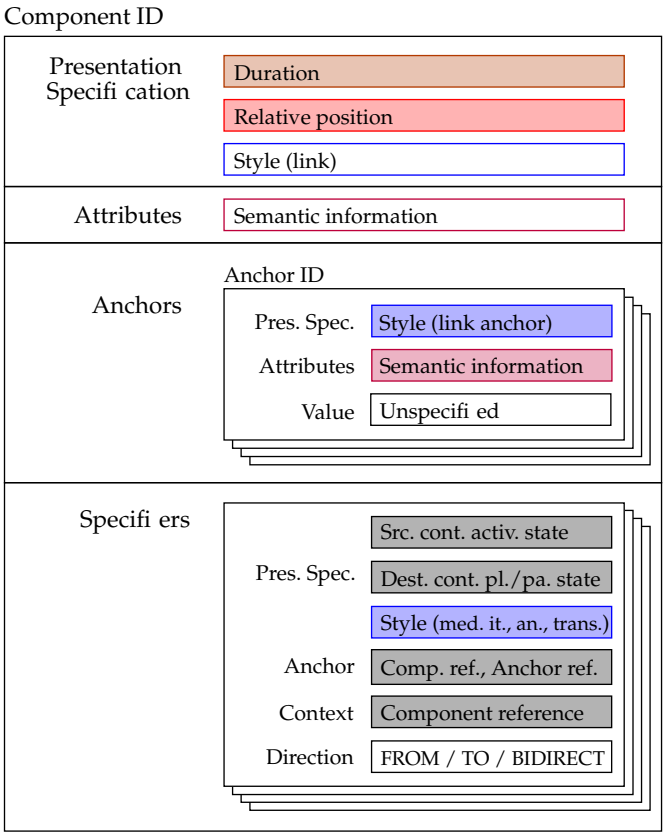
of the arc is the END of the source context and the destination of the arc is the START of the destination context.

- The *relative position* is the position of the destination context with respect to the source context, where their layout is not prespecifi ed via the channels, e.g. by being in different windows.
- The *style* information consists of a link style which can be used for creating displays of links.

Anchors

The *anchor* for a link component has the same structure as the anchor for the atomic and composite components.

- The *identifi er* and *attributes* are the same as for atomic or composite component anchors.



Changes from Dexter are greyed.

Figure 3.5. AHM link component.

- The *presentation specification* is a link anchor style that can be applied to a visual representation of link to link graph structures.
- The *anchor value* is outside the scope of the model.

Specifiers
A *specifier* stores the information for the (possibly multiple) ends of the link and is itself composed of a number of parts: a presentation specification, an anchor, a context and a direction.

- The *presentation specification* consists of a source context activation state, a destination context play/pause state and style information.
  The *source context activation state* determines the activation state for a specifier with direction FROM or BIDIRECT when it is part of the source context. The specifier context can remain active or DEACTIVATE when the link is followed. When it remains active the specifier context can CONTINUE to play or PAUSE.
  The *destination context play/pause state* determines the activation state for a specifier with direction TO or BIDIRECT when it is part of the destination context. The specifier context becomes active and can either PLAY or PAUSE. The possible styles are anchor, media item and transition special effect which can be applied when the link is followed. For example, for highlighting a source anchor to show that it has been selected, or for specifying the media item style of the destination context.
- The *anchor* specifies an end-point of the relationship represented by the link. It is given as a component reference/anchor reference pair. The component reference can refer to an atomic, composite or link component. The anchor reference may be omitted, with the interpretation that the complete component plays the role of the anchor.
- The *context* specifies the scope of the relationship at a link end and is given by a component reference. The context component reference is restricted to being an ancestor of the anchor component reference and may be equal to it. The context is thus guaranteed to contain the anchor. The context is further restricted to being an immediate child of an atemporal component, otherwise following the link may violate temporal relationships specified within a temporal composite.
- The *direction* specifies the direction of the relationship represented by the link and can be interpreted as a traversal direction. The values of direction are FROM, TO and BIDIRECT.

The transition information for the link consists of the link duration, the relative position and the special effect stored in the link specifier

Source and destination context
The *source context* of a link is the collection of specifier contexts that are active at the time the reader selects the link. The *destination context* is the collection of specifier contexts that are not part of the source context and that have direction

TO or BIDIRECT, i.e. the collection of specifier contexts that will be made active because of the selection of the link. Note that the source and destination contexts are runtime definitions. Note also that a specifier context may be in neither the source nor destination context, if it is inactive when the link is selected and has direction FROM.

Document requirements

The parts of the structure that require to be specified for a link component are at least one specifier. We do not go into the issues of dangling links, which are discussed in [GrTr94a]. The presentation specification (including transition information), attributes and anchors are optional. For each specifier the source context activation state, the context and direction are required. For each specifier with direction TO or BIDIRECT the destination context play/pause state is required. For each specifier with direction FROM or BIDIRECT the anchor is required. Specifier style is optional. The specifier context component reference is restricted to being the immediate child of an atemporal composite component when the direction is FROM or BIDIRECT and the source context activation is DEACTIVATE. The document requirements are summarised in Table 3.5.

Model elements			Required	Optional
Link Component	Presentation Specification	Duration		*
		Relative position[a]		*
		Style		*
	Attributes			*
	Anchors			*
		Anchor ID	*	
		Pres. Spec.		*
		Attributes		*
		Value	*	
	Specifiers		*[b]	
		Source cont. activ.	*[c]	
		Dest. cont. pl./pa.	*[d]	
		Style		*
		Anchor	*[c]	
		Context	*	
		Direction	*	

TABLE 3.5. Link component and required/optional document specifications

a. Valid only if destination context is in a different window from the source context.
b. At least one is required.
c. Required only for specifier directions FROM and BIDIRECT.
d. Required only for specifier directions TO and BIDIRECT.

Discussion
The AHM link component is based on the Dexter link component where the presentation specification is further specified to include style and transition information and the specifier is extended to include context information.

A link can be thought of as specifying a temporal composite with source and destination contexts as child components. The difference is that the temporal relations among the child components are brought into play only at runtime when the link is followed.

Link behaviour is discussed in [LeSc94] and [Hala88], where the authors point out that Dexter is in some cases too restrictive (for example by specifying the traversal behaviour) and that the link behaviour should be encapsulated in the storage layer rather than embedded in the particular hypermedia system. It is this type of information that we capture with the combination of context and presentation specification per specifier. Link context is semantic, delimiting the scope of the relationship represented by the link. This is interpreted at runtime as the scope of the presentation which is affected on following the link.

Link specifier context is restricted to being an ancestor of the anchor component reference, so that the anchor is guaranteed to be within (a descendant of) the context. If the anchor were not part of the context then the system would have no way of controlling the activation and deactivation of the part of the presentation that contains the anchor.

A link specifier can act as the source of a link when the specifier has direction FROM or BIDIRECT. When the source context activation state is DEACTIVATE, the context of a source link specifier is restricted to being the immediate child of an atemporal component. We impose this restriction to ensure that prespecified timing relations are not violated. For example, if the specifier context has a temporal composite as a parent then deactivating the source context would leave only part of a temporal composite playing.

Similar issues arise with link destination specifiers. A link specifier can act as the destination of a link when the specifier has direction TO or BIDIRECT. Problems arise when there are synchronization arcs from outside the destination context to within. In such cases, when you play the destination context only the synchronization relationships specified within the destination itself are applicable. Any relations outside the composite are ignored. We thus recommend that the destination context be an immediate child of an atemporal composite, but do not require it in the model.

The link's presentation specification includes the transition duration and relative position applied at runtime when traversing the link. This transition information is stored in the link's presentation specification rather than in each of the specifiers since it pertains to the relationship represented by the link. It means, however, that different durations for the potential multiple link ends cannot be specified. We have had no experience with an implementation of transitions and

links with more than two specifiers, so are unable to say whether the model as it stands will be sufficient or not.

The duration of the link transition is specified using a synchronization arc since it is not the duration of a single component but is the temporal relationship between the source context and the destination context. Additional information, such as the synchronization type, is thus desirable.

The link style is applicable to a display of links in a network diagram, for example as implemented in existing hypertext systems [Hala88], [HKRC92].

In Dexter the specifier direction may have the value NONE where, as is pointed out in [GrTr94a], the meaning of NONE is unclear. Our preference for the interpretation of the direction is semantic, that is, it should specify the direction of the semantic relationship among the link-end components rather than a traversal direction. For example, component A "is an example of" component B. The traversal direction is then a presentation property associated with the link type. In the example, the link can be followed in either direction while the relationship is asymmetric. Given that both the semantic and navigational interpretations of direction NONE are not clear, we thus limit the set of directions to FROM, TO and BIDIRECT.

The *anchors* in a link allow other links to refer to it from their specifiers, allowing, e.g., knowledge structures to be created. While we consider link anchors to be outside the scope of the AHM, where linking is interpreted as a relationship that can be presented in terms of navigating among multimedia presentations, others use this construct [GrTr94a], and we see no reason for explicitly excluding it from the document model. The AHM link anchor extends the Dexter link anchor to include attributes and presentation specifications. The link anchor value is outside the scope of both the AHM and Dexter models. Attributes associated with a link anchor can be used to express the semantic relationship being represented by the link to link structure. The presentation specification can be used to specify a style for displaying a representation of the link anchors in a network diagram. We do not discuss this further.

3.3 A runtime perspective of the model

The previous section defines the Amsterdam hypermedia model in terms of the components within the model. This section goes through the presentation aspects expressible within the model and shows how information stored within different components of the model can be combined together at runtime in the final presentation. These aspects are: temporal layout, spatial layout, styles and activation state.

3.3.1 Temporal layout

Temporal information occurs explicitly and implicitly throughout the components in the model. Explicitly in the presentation specifications:

- atomic component duration,
- anchor value duration and start time of an atomic component of a non-continuous media type,
- temporal composite duration,
- synchronization arc scheduling interval,
- link duration,

and implicitly in:

- the content of an atomic component of a continuous media type,
- an anchor value of an atomic component of a continuous media type,
- the children and synchronization arcs of a temporal composite component.

The content of an atomic component, of a continuous media type, has its own intrinsic duration. This can be used as the duration of the atomic component, and stored in the duration specification of the component, or another duration can be specified. Two "contradictory" durations have no consequences for the rest of the model, since the specified duration is the one used, but a system presenting the content would need to ensure that the playing of the content lasted as long as the specified duration, for example by stopping the playing before the end of the content is reached or by playing the content faster.

An anchor of an atomic component has a start time and a duration. When the content is of a continuous media type the start time and duration are calculable from the anchor value specification. For a non-continuous media type these can be specified. The model does not require a start time or duration for the anchor, although this is required for an atomic component. The temporal information for the anchor is not referred to from elsewhere in the model, so does not have to be stated explicitly.

In the AHM temporal composition is "bottom-up" composition, that is, the duration of a temporal composite component is derived from the timing of its (content-containing) children and the related synchronization arcs, e.g. Fig. 3.6. There may also be an explicitly specified duration for the composite, in which case the specified duration should be used in the rest of the model (for example for inclusion in other temporal composites). The player software should ensure

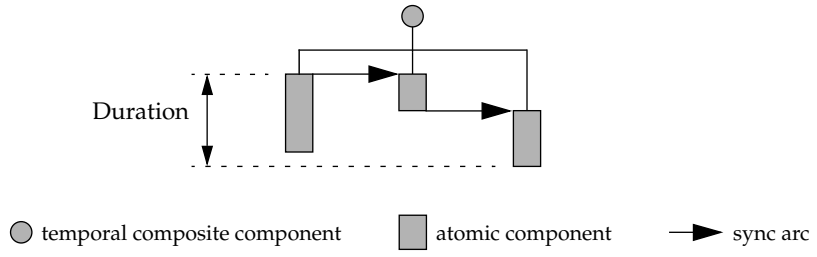


Figure 3.6. Duration of temporal composite component

that the playing of the children of the composite lasts as long as the specified duration, for example by taking appropriate scaling or cropping actions.

It is beyond the scope of the model to resolve the interpretation of inconsistent temporal relations given by the synchronization arcs among the descendants of a temporal composite.

3.3.2 Spatial layout

Spatial information occurs explicitly and implicitly throughout the components in the model. Explicitly in the presentation specifications:

- channel position and extent,
- atomic component position and extent,
- link relative position,

and implicitly in:

- the channel reference in a channel (position and extent),
- the channel reference in an atomic component (position and extent),
- an anchor value in an atomic component of a visual media type (position and extent),
- the content in an atomic component of a visual media type (extent).

The extent of the content of an atomic component can be found in three places in the document model. Firstly, if the content is of a visual media type it may have an intrinsic extent. Secondly, the extent can be specified in the atomic component presentation specification. This may have an absolute value, or may be given in terms of the extent of the content or in terms of the channel extent. The exact terms of specification fall outside the scope of the model. Thirdly, the channel associated with the atomic component has an extent.

The position of the content can be found in two places in the document model. Firstly, in the atomic component as a position with respect to the channel reference. This can be in terms of the position and extent of the channel, e.g. centre left-right and top-bottom with respect to the channel's position and extent. Secondly, in the channel as a position with respect to the channel reference's extent and position. Example position and extents are shown in Fig. 3.7.

The atomic component layout is specified in terms of the channel. The channel can provide default layout specifications, such as "align to top, center left-right, scale to fill", which can be overridden by the specifications in the atomic component, e.g. "center top-bottom". The link specifier layout can be used when the source and destination are in different windows.

Note that the position and extent specifications throughout the model may vary with time.

In contrast to the temporal specification, the spatial composition in AHM is "top-down", that is, the extent of a channel is specified in terms of its channel reference, Fig. 3.7. The top of the hierarchy is a system defined window. This allows the layout to be determined independently of the extents of the atomic components, and, more importantly, allows consistent layouts to be designed

with no knowledge of the content that is to be played and without imposing restrictions on the extent of the content. Another advantage is that one particular document can be played back on a variety of screen sizes without having to alter the document specifi cation in any way It is the system window which can be scaled and the rest of the layout specifi cations follow

3.3.3 Temporal and spatial layout combined

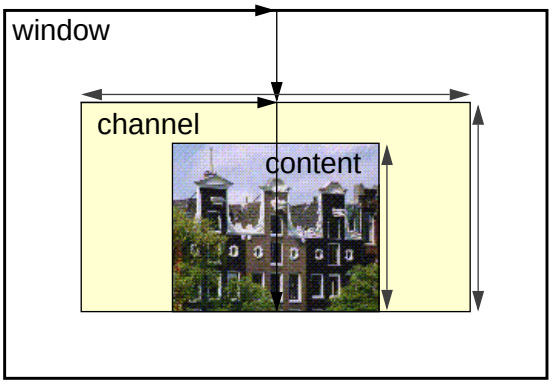
The layout structure of a presentation is independent of its temporal structure. Both, however, are needed for the complete description of a presentation. The temporal hierarchy and the layout hierarchy meet at the atomic components, illustrated in Fig. 3.8. The durations of the atomic components form the building blocks of the presentation's timing. The channels, defi ned in terms of system windows, form the layout hierarchy. A piece of content is displayed in relation to the channel associated with the atomic component, from the time determined by the temporal hierarchy for the duration specifi ed in the atomic component.

3.3.4 Styles

The styles that occur throughout the model are media item style, anchor style and transition special effect. Link style and link anchor style are also mentioned, but we do not discuss these further. Further specifi cation of the styles themselves falls outside the scope of the model, although examples were provided in Chapter 2.

Media item and transition styles can be found in four places in the presentation specifi cations:

- channel,
- atomic component,
- composite component,



Channel in terms of window:
position centred left-right,
position top at height 25%,
height 60%,
width 70%.

Content in terms of channel:
position centred left-right,
position aligned bottom,
height 80%,
aspect ratio preserved.

Figure 3.7. Spatial layout

- link specifi er

Anchor style can be found in six places in the presentation specifi cations:

- channel,
- atomic component,
- atomic component anchor,
- composite component,
- composite component anchor,
- link specifi er

The application of styles in increasing order of override is the following:

- channel,
- atomic component then atomic component anchor (for anchor style only),
- composite component then composite component anchor (for anchor style only),
- link specifi er

The atomic component style overrides the channel style, since the channel gives only a default to save multiple specifi cation for each atomic component. The composite component style overrides the atomic style, since the same atomic may be included in a number of composites where the context requires a different style, for example background colour. Similarly, the link specifi er style over-

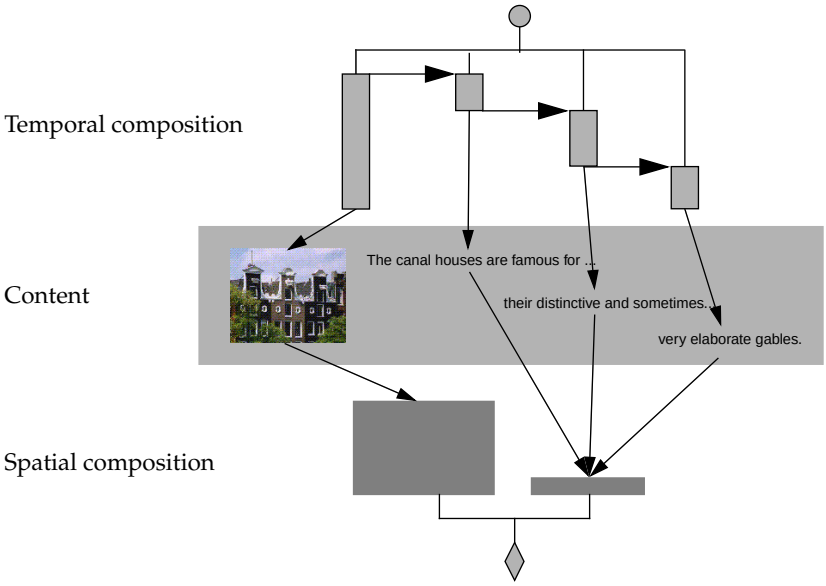


Figure 3.8. Temporal hierarchy meets spatial hierarchy

rides the composite component style since the same composite used in the destination context of a number of links may require different style.

We repeat Figure 12 from Chapter 2 to illustrate where in the model the different styles are stored, Fig. 3.9. Anchor markers are shown as boxed markers for clarity, whereas in an active system they may be indicated more dynamically, e.g. using change of mouse cursor shape.

(i) Source anchor is displayed.
Anchor style is given in channel.

(ii) User selects anchor.

(iii) Source anchor highlights.
(Anchor border thickens and
text background changes colour.)
Anchor highlight style is given in
FROM link specifi er

(iv) Source context dissolves into
destination context.
Transition style is given in source and
destination link specifi ers.

(v) Destination context displayed.
(Destination anchor not highlighted.)

Veel van de huizen en ...

Figure 3.9. Anchor and transition styles on following a link

3.3.5 Activation state

Activation has two aspects— whether an item is active or not, and whether it is traversing its own timeline. Anchor marker selections can be made when the component is active.

Activation state is recorded in two places in the document model:

- with each child in the presentation specification of an atemporal composite component,
- as part of the source context activation state of the link specifier

Play/pause state is recorded in three places in the document model:

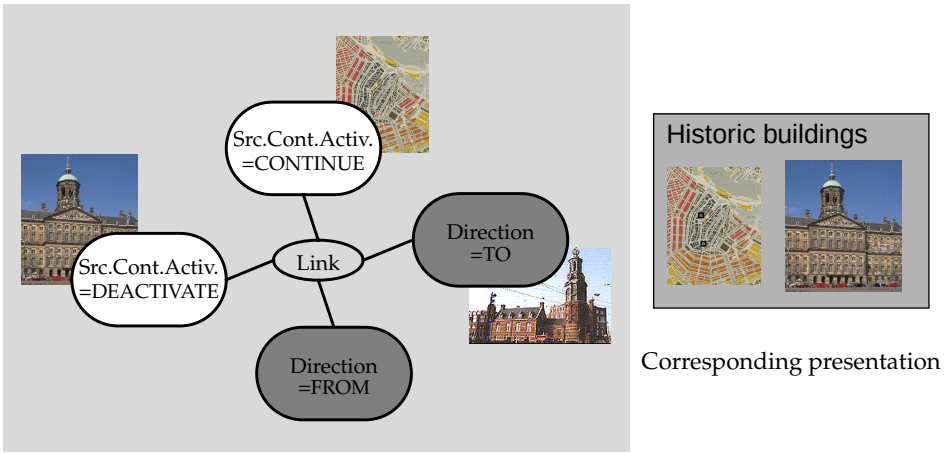
- with each child in the presentation specification of an atemporal composite component.
- as part of the source context activation state of the link specifier,
- the destination context play/pause state of the link specifier

Each child of an atemporal composite has an initial activation state. A presentation thus begins with the children of an atemporal component which are marked as active. Each active child starts up as playing or as paused. The activation state is stored in the atemporal composite rather than with each child, since each child may be included in other composites with different activation states, Fig. 3.3. The activation state changes according to links that are followed. The source context behaves as specified in the link specifiers, according to the source context activation state—DEACTIVATE/CONTINUE/PAUSE. The destination context defines the presentations that become active, i.e. those not in the source context and with direction TO or BIDIRECT. The destination context play/pause state specifies whether the specifier context should play or pause.

We first describe the activation state of the simple case of following a link with two specifiers, one with direction FROM and the other TO. The specifier with direction FROM is active and the reader selects the corresponding anchor marker. The source context is continued, paused or deactivated, depending on the source context activation flag. The destination context is activated and is played or paused depending on the destination context play/pause flag.

We now describe the case for a link with multiple specifiers. An example is given in Fig. 3.10. The reader selects one of the active anchor markers, in the example the lower anchor marker on the map. Since the reader was able to select it, the component referenced by the specifier must be active, so the specifier is by definition in the source context. Other specifiers whose context is active are also in the source context, for example the picture of the palace. Each specifier context in the source context with direction FROM or BIDIRECT is deactivated, continued or paused depending on the value of the source context activation flag. In the example the map is continued and the palace is deactivated. If the specifier context in the source context has direction TO then the specifier context is continued. The destination context is the set of non-active specifiers in the link with direction TO or BIDIRECT. In the example the picture of the tower is shown. Each specifier context in the destination context is made active, taking into account the value of the play/pause state flag.

Before actioning the link:



After actioning the link:

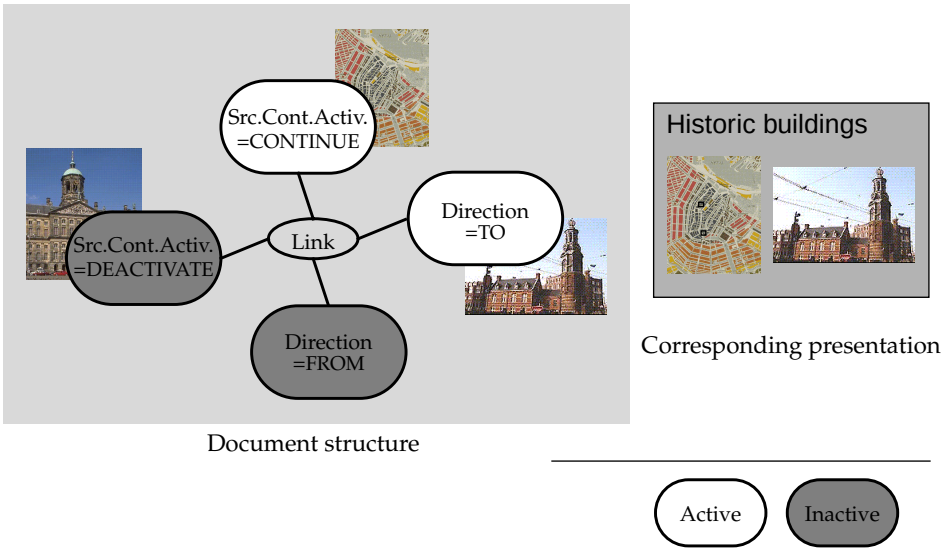


Figure 3.10. Change in activation state on following a link

3.4 Summary and discussion of the model

Having described the model components in Section 3.2 and discussed the presentation specifi cations in Section3.3 we provide summaries and a discussion in this section. A summary of the model and the requirements for a document to conform to the model are given in Section 3.4.1 and in Table 3.6. To demonstrate that the document model meets the requirements stated in Chapter 2, the correspondence between these and the AHM elements is given in Section 3.4.2 and summarised in Table 3.7. A discussion of the model is given in Section 3.4.3.

3.4.1 Summary of AHM

A summary of the AHM is given as a diagrammatic impression in Fig. 3.11. This shows the combination of time and structure in one document model and is based on Figures 13 and 15 in Chapter 2.

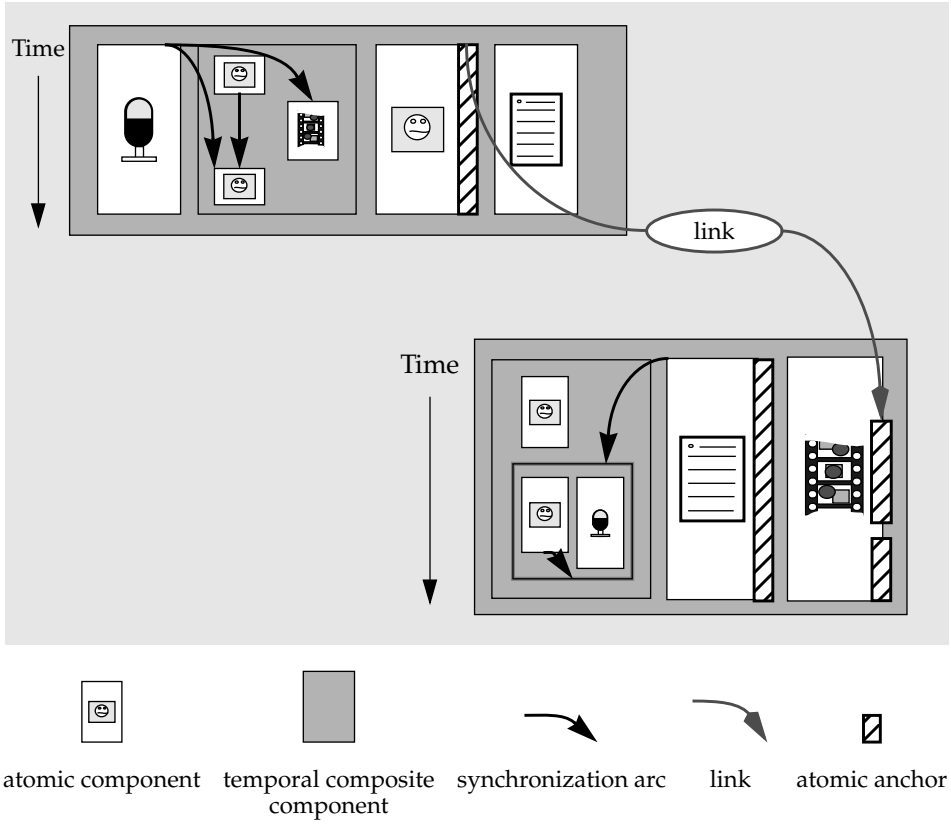


Figure 3.11. Amsterdam hypermedia model overview.

The main components of the model are:
- media item,
- channel (not shown),
- atomic component and atomic anchor,
- temporal composite component including synchronization arcs,
- atemporal composite component (not shown),
- composite anchors and
- link.

Structure elements from the Dexter model have been extended to include explicit temporal and spatial information, and anchors for composite components have been defined explicitly. Synchronization arc and channel structures from the CMIF model have been included and are described explicitly.

A more detailed summary of the model is given in Table 3.6, where the requirements for a document conforming to the model are also summarised. References back to the section where each requirement was first stated are also included. Each component requires an identifier, not shown in the table.

A formal description of the model in the Object Z language is given in [OsEl97]. This is included as Appendix 2 of this thesis as a supplement to the definition provided here.

Model elements			Required	Optional	Stated in section
Channel	Presentation Specification	Channel ref.	*[a]		3.2.2 Channel
		Position & extent	*[a]		
		Style		*	
	Attributes			*	3.2.2 Channel
	Media type		*		3.2.2 Channel
Atomic Component	Presentation Specification	Duration	*[b]		3.2.3 Atomic Component
		Channel ref.	*		
		Position		*[a]	
		Extent		*[a]	
		Style		*	
	Attributes			*	3.2.3 Atomic Component
	Anchors			*	3.2.3 Atomic Component
		Anchor ID	*		
		Pres. Spec.		*	
		Attributes		*	
		Value	*[c]		
	Content	Media item ref.	*[d]		3.2.1 Media item,
		Data-dep. spec.		*	3.2.3 Atomic Component

TABLE 3.6. AHM elements and required/optional document specifications

Model elements			Required	Optional	Stated in section
Temporal Composite Component	Presentation Specifi cation	Duration		*	Temporal composite presentation specifi cation
		Sync. arcs	*		
		Style		*	
	Attributes			*	3.2.4 Composite components
	Anchors			*	3.2.4 Composite components
		Anchor ID	*		
		Pres. Spec.		*	
		Attributes		*	
		List of anchors	*		
	Children	Comp. ref.	*[e]		3.2.4 Composite components
Atemporal Composite Component	Presentation Specifi cation	Initial activ. state	*[f]		Atemporal composite presentation specifi cation
		Play/pause	*[f]		
		Style		*	
	Attributes		As for Temporal Composite		
	Anchors		As for Temporal Composite		
	Children		As for Temporal Composite		
Link Component	Presentation Specifi cation	Duration		*	3.2.5 Link component
		Relative position[g]		*	
		Style		*	
	Attributes			*	3.2.5 Link component
	Anchors			*	3.2.5 Link component
		Anchor ID	*		
		Pres. Spec.		*	
		Attributes		*	
		Value	*		
	Specifi ers		*[f]		3.2.5 Link component
		Source cont. activ.	*[h]		
		Dest. cont. pl./pa.	*[i]		
		Style		*	
		Anchor	*[j]		
		Context	*		
		Direction	*		

TABLE 3.6. AHM elements and required/optional document specifi cations

a. For a visual media type.
b. The duration may be indefi nite or unpredictable.
c. Each anchor value specifi cation is restricted to being within the content, i.e. within the data-dependent specifi cation of the media item reference.
d. The intrinsic duration and spatial extent are also known.
e. At least one is required.
f. One per child is required.
g. Valid only if destination context is in a different window from the source context.
h. Required only for specifi er directions FROM and BIDIRECT.
i. Required only for specifi er directions TO and BIDIRECT.
j. Optional if the specifi er is of direction TO.

3.4.2 Showing the AHM meets the requirements

Table 3.7 gives a summary of the requirements stated in Chapter 2 and shows which parts of the model satisfy the requirement. Most of these are straightforward, in that each requirement is satisfied using one, or part of one, component. Two, however, are satisfied through a combination of different parts of the model: Allen's relations, and the specification of a time axis.

3.4.2.1 Specification of time axis

The model includes no explicit timeline in a temporal composite component. The time axis is instead calculated on the basis of the temporal composition structure, including the duration of the descendant components and the synchronization arcs specifying constraints among the components.

3.4.2.2 Allen's temporal relations

Temporal relations between two components (atomic or temporal composite) are stored as synchronization arcs. Complete relative timing requires the support of all of Allen's temporal relations [Alle83]. We show that synchronization arcs can express these in Fig. 3.12.

We have thus demonstrated that the Amsterdam Hypermedia Model satisfies the requirements for a hypermedia document model as stated in Chapter 2.

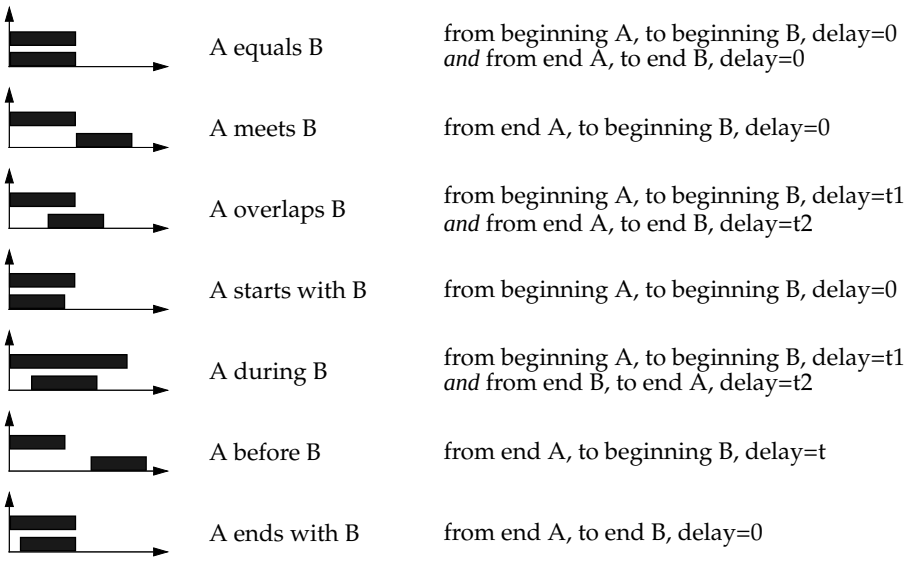


Figure 3.12. The Allen time relations expressed as synchronization arcs.

Dexter layer	Document feature	Requirement	Satisfied in
With. comp.	Media items	temporal and spatial dimensions	3.2.1 Media item
Anchoring	Ref. to part of media item	data-dependent specifi cation of part of media item	3.2.3 Atomic Component, Anchors and Content
Storage layer	Instance of media item	reference to (part of) media item, data format, duration, start time, extent, pos., asp. rat., orien., Z-order, style (media item, anchor, transition) start points for links, semantic attributes	3.2.3 Atomic Component, Content 3.2.2 Channel 3.2.3 Atomic Component, Presentation specifi cation 3.2.3 Atomic Component, Anchors 3.2.3 Atomic Component
	Composition	temporal composition, spatial composition, space/time independ. composition anchor composition	3.2.4 Composite components, Temporal 3.2.2 Channel 3.2.4 Composite components, Atemporal 3.2.4 Composite components, Anchors
	Linking	source and destination anchor, source and destination context, transition (duration and special effect)	3.2.5 Link component, Specifi ers 3.2.5 Link component, Specifi ers 3.2.5 Link component, Presentation specifi cation
	Semantic attributes	associate with: anchors, instances, compositions of anchors, compositions of intances, and links	3.2.3 Atomic Component, Anchors 3.2.3 Atomic Component 3.2.4 Composite components, Anchors 3.2.4 Composite components 3.2.5 Link component
Presentation Specifi cations	Temporal layout	time axis, start time, duration, scaling, Allen's relations, link transition duration	3.4.2.1 Specifi cation of time axis 3.2.4 Composite components, Temporal 3.2.1 Media item, 3.2.3 Atomic Component, 3.2.4 Composite components, Temporal 3.2.4 Composite components, Temporal 3.4.2.2 Allen's temporal relations 3.2.5 Link component, Presentation spec.
	Spatial layout	space axis, position, possibly changing w.r.t. time, extent, scaling, orientation, aspect ratio, Z-order, link transition spatial relation	3.2.2 Channel 3.2.2 Channel, 3.2.3 Atomic Component (not w.r.t. time) 3.2.2 Channel, 3.2.3 Atomic Component 3.2.2 Channel, 3.2.3 Atomic Component 3.2.2 Channel, 3.2.3 Atomic Component 3.2.2 Channel, 3.2.3 Atomic Component 3.2.2 Channel, 3.2.3 Atomic Component 3.2.5 Link component, Presentation spec.
	Styles	associate with anchors, instances, compositions of anchors, compositions of intances, and links	3.2.3 Atomic Component, Anchors 3.2.3 Atomic Component 3.2.4 Composite components, Anchors 3.2.4 Composite components 3.2.5 Link component
	Activation state	initial activation state change in activation state	3.2.4 Composite components, Atemporal 3.2.5 Link component, Specifi ers

TABLE 3.7. Summary of requirements for a hypermedia document model

3.4.3 Discussion of the AHM

We have already discussed the pros and cons of the individual components of the model in the previous discussion subsections. This section discusses the model as a whole, with emphasis on potential improvements and extensions.

The model is based predominantly on the Dexter model, and thus inherits a number of the limitations that apply to the model, e.g., those discussed in [Hala88], [LeSc94]. Some of these are solved by others, in which case we would wish to inherit the solution. We do not intend to resolve them, since the goal of the AHM was not to improve a hypertext model, but to propose a hypermedia model.

A component reference in the AHM is not restricted to any notion of document boundary. The only requirement is that the player has access to the component. For example, if the component reference is given using a URL the player would have to support the HTTP protocol. The main consequence of this is that there is no guaranteed bound to the number of components making up the hypermedia presentation. This is not a problem for activating and deactivating parts of multimedia presentations, but would be a problem if a representation of the link graph was desired.

Channels

Channels are introduced in the model as a means of specifying spatial layout and of allowing the specifi cation of style defaults for atomic components. The introduction of the channel construct in CMIF [BuRL91] was to provide high level control of runtime resource allocation. These two roles remain mixed and somewhat implicit in the AHM. In particular, the channel could be split into a more runtime resource-oriented construct which refers to a layout construct. The layout construct could independently specify layout which could be referred to from multiple channels, particularly useful for items of different media types, e.g. image and video, which are intended to use the same screen space.

While the model does not restrict the movement of a channel with respect to its channel reference with time, this would violate the resource function of the channel.

A limitation of the use of channels for layout is that items cannot be positioned with respect to the content of other items. This is suffi cient for a wide range of presentations, but not, for example, for the overlay of maps of a city taken from different periods to show how the city expanded. Here a part of a map requires to be overlaid accurately onto a part of another map. Scaling in terms of the map dimensions as a whole is insuffi cient. Introducing spatial relations among anchors, i.e. the model construct for referring directly to content, is a potential solution. If layout with respect to content were introduced then the channels would lose their layout independence, thus requiring a more complex implementation.

Anchors
Anchors in the model are based on data-dependent descriptions of the content of atomic components. Since the model makes the dimensions of time and space explicit, any part of the presentation, whether a temporal composite or an individual atomic component, can be specified in terms of the temporal and spatial extents of the presentation. For example, an anchor of a component could be defined which starts at 20% of the duration of the component and continues until 25% of the duration. This type of anchor specification could be used in the same way as the current anchor as the end of a link or a synchronization arc. We have chosen not to include this type of anchor in the model, since the semantics of the anchor value are not clear, and the difficulties of guaranteeing a valid anchor value after editing a media item are even greater than those for data-dependent anchor value specifications.

Link component
While we noted in the link component discussion that we prefer the notion of direction for a link to be based on semantics rather than on traversal, we use link specifier direction in the model as a traversal direction. An extension to the model would be to replace the single direction with a semantic direction and a traversal direction, so that the behaviour of the link could be independent of its semantics. Up until now our own use of the model has been more presentation based than semantically based so we have not yet experienced the need for two distinct directions.

A link has a single duration and spatial relation, whereas it is probably more appropriate to assign these per specifier, so that the timing and placement of the, possibly multiple, destination contexts of the link can be individually specified. The duration would be specified with respect to the time of the user interaction and the spatial relation with respect to the anchor marker selected.

Presentation Specifications
The figures of the model components in this section give the impression that we have specified the presentation specification of all the components completely While we have divided the original Dexter presentation specification into temporal, spatial, style and activation information we do not intend to imply that this is all that a presentation specification can contain. Just as an application is able to define its own semantic attributes and styles, the presentation specification can be extended for application dependent purposes. The AHM states that certain aspects of the presentation specifications occur throughout the model, in particular the temporal and spatial information, which are interdependent. An example of an additional presentation specification is whether an anchor's marker should be displayed or not.

Temporal layout
The model includes no explicit timeline in a temporal composite component. Without an explicit timeline we cannot specify changes of tempo within the

presentation. We are able, in the current model, to specify the tempo of individual components, but *accelerando* and *ritardando* over a temporal composite cannot be specified. In order to incorporate this information, one could include a timeline in the component specific presentation specification for a temporal composite along with specifications of the tempo along the timeline.

Time is treated within the model purely as relative. The initial start time of a document is when the reader starts up a session and all other times are relative to this start time or relative to when the reader follows a link. It would be useful to allow absolute times, such as 18:00:00 MET, to be included in the model, but this would require the development of activation mechanisms other than those already provided by linking.

Spatial layout
While a composite component may refer to a number of atomic components each with their own spatial layout specifications, it might be the case that for a particular composite the author requires a different layout. An example of this is the table top composite in [Grøn94]. The model does not allow the specification of layout information in the composite's own presentation specification. If the model were to include layout information in a composite then an extra level of override would be needed—the channel specifies the default layout, the atomic component overrides it, and the encompassing composite component would take precedence.

Activation state
The current activation states included in the model are whether the presentation is active or not and whether an active presentation is playing or paused. An additional state is possible which distinguishes whether the components are visible or invisible. The assumption is made within the model that if a component is active that it is also visible. This does not, however, allow synchronized streams to be turned on and off while they are playing without losing their current play state. Useful, for example, for playing a video with synchronized commentaries in different languages where the reader is able to select which language to see and/or hear. If a component were to be turned off, in the current model, it is by definition deactivated. Turning it on again would force it to be played from the beginning again. With the addition of a visible/invisible state the component could be turned off, but still continue to play. This would require a mechanism for stating the initial visible state and how this changes via user interaction, akin to that for the play/pause state already defined in the model.

3.5 Implicit document models of existing systems expressed in AHM

Chapter 2 motivated the requirements for a hypermedia document model using a simple presentation. This example provided a lower bound on the features necessary for a document model. This left open the question of whether the

requirements would be sufficient for describing presentations created by a broad range of systems. In this section we show that the AHM is able to describe the presentations created by a range of existing systems. We thus demonstrate that the AHM is a valid and useful model of hypermedia.

The systems whose documents we describe with the AHM are selected as being representative of hypertext systems, multimedia systems and, although there are to date fewer of these, hypermedia systems. For each system we give a description of its document model in its own terms, and then a brief description in terms of the AHM. A summary of the systems' documents in terms of the AHM is given in Table 3.8. Note that in this section we can only categorise elements of the implicit document models if they have been reported in the referenced articles. The summaries in each section and the overview in Table 3.8 can thus state only the presence of elements and not the absence of elements. Hence the occurrences of "yes" in the table and the absence of "no".

3.5.1 Hypertext

3.5.1.1 Intermedia

Intermedia [HKRC92] has a database of nodes of media types text, graphics, music and animation. A node of any of the media types can have anchors. A link connects two anchors, and any anchor can be the start or end of multiple links. Sets of links are stored separately as webs. A number of webs can apply to the same set of nodes. Active destinations are introduced using active anchors [PaYS90], i.e. anchors that contain a flag specifying whether the destination node should play or not. This means that while Intermedia does not incorporate time within the document model, it is able to control whether a continuous medium is played on activation or displayed only statically. The original system influenced the development of the Dexter model, and can be described with the model.

In AHM terms

In terms of the AHM, an atomic component has a presentation specification, anchors and content. Spatial information for the extent and position of the window in which the content is displayed is possibly recorded. Each anchor has an identifier, a value and a (visual) marker that is dependent on the application displaying the media item. That an atomic component, of a continuous media type, should start playing on arrival can be recorded as part of the presentation specification of the specifier of the link.

Intermedia does not support composition of atomic components. A Web is a composition of links. A link component is composed of two specifiers each with an anchor and presentation specification. There is no additional link component information. Each specifier has the implicit direction BIDIRECT, since either anchor can play the role of the beginning or end of the link. Any anchor may be referenced by multiple links.

3.5.1.2 Guide

The Guide hypertext system [OWL90] supports the creation of a number of structural objects termed reference buttons, expansion buttons, note buttons, and command buttons. Reference buttons are links to a new position in the same document or in another document, expansion buttons expand information inline, note buttons display additional information in a temporary window and command buttons execute scripts.

Guide has a number of composition types. A type of atemporal composition is the frame which allows one document to be divided into a number of subdocuments. Each subdocument uses the same display screen estate. An expansion button groups atomic components together and initially displays only the content of the expansion button. When the reader selects an anchor marker the content belonging to the corresponding child is displayed in the text-flow of the parent. The AHM has no equivalent composition model, since text flow is not part of a spatial/temporal model, although including text within the flow does require spatial position and extent information. Guide also includes an atemporal/aspatial composition for the inclusion of the destinations of note buttons. These are displayed in separate windows but are stored as part of the document.

The presentation specifications for a number of styles are hard-wired into the system itself, e.g. the destination of a note button always appears in a separate window. Other styles can be specified, e.g. a number of anchor styles are supported by the system. A source anchor marker can be highlighted by changing the cursor when it is over it, the style can change when the reader clicks and the destination anchor marker can be highlighted.

A Guide link has context. A number of expansion buttons can have an enclosing "group" object which specifies that when any child is expanded that any open child is closed.

In AHM terms

An atomic component (a button) consists of anchors and content. Each anchor has an ID and a value. The content can be text, graphics, video or script.

A composite component, an expansion button, is a hierarchical collection of atomic and composite components. This is an atemporal composition where the child is displayed within the text flow of the parent. A composition of expansion buttons, the group, is used to indicate the source context for its children. The frame is an atemporal composition grouping subdocuments which share the same spatial layout. Composition is always by inclusion and not by reference.

A link is not a separate component but is included within the anchor information in an atomic component. It consists of a source and destination specifier. A source specifier (direction FROM) consists of a source context activation specification, an anchor style, an anchor and a context. The anchor style is the style for the cursor when it is over the source anchor. The context is deduced from the document structure. A destination specifier (direction TO) consists of an anchor

style and an anchor. The anchor style is where the destination anchor will be shown, e.g. at the top of the window, and whether the anchor marker should highlight.

3.5.1.3 Microcosm

The philosophy of the Microcosm system [HaDH96] is to allow the creation of links among documents that are not necessarily part of a hypertext authoring environment. This requires the specification of anchors and links externally to the documents being linked. The Microcosm designers also wanted to provide links without forcing the author to create each one separately by hand. They thus provided the facility for creating links from any occurrence of a word without requiring the author to specify its position in the document. These are termed generic links.

A linkbase is collection of links. Several linkbases can refer to the same sets of documents.

In AHM terms

An atomic component consists of semantic attributes and content. The semantic attributes include the name of the file's author, and any number of keywords or author-defined attributes. The content is a reference to a file and a logical type for the file.

Anchors are not contained within an atomic component but within a link specifier. Each anchor has a value and semantic attributes.

Composition can apply to anchors and is implicit within the generic link component. Composition can also apply to links.

A link has a source and a destination specifier. The source specifier (direction FROM) is a description of one or more occurrences of an anchor within an atomic component. The destination specifier (direction TO) is a single anchor in an atomic component.

3.5.1.4 HTML

A document conforming to an HTML [Ragg97] specification contains a small number of hypertext-specific objects. The basic notion is of a single text flow including presentation information within which anchors can be defined. Anchors can contain the specification of another HTML document or a marker within the same or another HTML document. When the reader selects an anchor marker the link is followed. The browser highlights the anchor markers in different colours allowing the reader to see which link destinations have already been visited.

There is as yet no structural composition in HTML, where the "root atemporal composite" is the complete collection of documents on the World Wide Web. A form of spatial composition is included in the HTML 4.0 proposal [W3C97] called a frame. This divides up the browser window area and allows multiple text-flows to be displayed in the different areas.

In AHM terms
An atomic component (a single file) consists of a presentation specification, anchors and content. The presentation specifications consist of text-flow positioning information and text, anchor and background style information. Anchors consist of an optional identifier, a presentation specification, content and a specification of the link destination.

Spatial composition is possible using the frame construct.

Links are single source, single destination and unidirectional. The direction is from the source anchor in which the destination information is stored.

3.5.2 Multimedia

3.5.2.1 Athena Muse

The Athena Muse system [HoSA89] describes hypermedia documents through the use of a directed graph of packages, where a package can be considered as a small multimedia presentation consisting of text, video and graphics media items. The packages are nodes in a directed graph, with arcs representing possible transitions between the packages. The arcs can be used to represent hypermedia links, where cross references are fired by sending activation signals from one package to another. The network is built on a concept structure with links between high-level abstractions, although how these are translated to the package level is not described.

Each package can be described in an N-dimensional space, where, for the case of multimedia, three dimensions are sufficient for describing temporal and spatial layout. An example is the use of a timeline for attaching subtitles to a video sequence. This is done by introducing a timeline and synchronizing both the video sequence and the subtitles with respect to the timeline.

The structuring of a package does not continue down to the sub-package level, neither does it extend above the package level to group sets of packages together other than as being part of the overall application.

In AHM terms
An atomic component (a package) consists of a presentation specification, anchors, and content. The presentation specification refers to a channel and specifies a duration. It is unclear whether the spatial layout information is specified per channel or per component. The content is a reference to a media item (text, video, graphics). Anchor values can be specified within the content, in particular images.

Composition is restricted to a one-level temporal composition of atomic components and atemporal composition of temporal composite components.

There is no explicit link component, but this can be deduced from the transition arcs between the atomic components. These specify the destination of the link and the temporal and style properties of the transition.

3.5.2.2 Eventor

The Eventor system [ENKY94] has four types of objects: basic, time, composite and input. Each has common attributes such as visual forms and sync data. A visual form has information about its figures and the relations with other data objects. Sync data has information about temporal and spatial synchronizations. Basic objects point to the data and input objects have a set of input ports.

In AHM terms

An atomic component consists of a presentation specification and content. The presentation specification consists of spatial layout (extent and position) and duration. The extent and position of the object can vary with time. The content is a reference to a media item (audio, video, image, text). Anchoring is unsupported in the system.

A composite component is similar to the atomic component where instead of the content it has a list of child objects.

Link components do not exist explicitly and traversals are implemented using a user interaction builder. Jumps can be from an atomic component.

The model also includes a time object, which can be considered an atomic component without any associated content, but with a duration.

3.5.2.3 Integrator

In the Integrator [SFHS91] media items can be sequenced in virtual time, and then mapped for display onto real time through the use of a score. The data items in the multimedia presentation are placed on individual tracks (analogous to staves in a musical score and similar to channels). Timing and synchronization of multimedia items within a single track are determined by their horizontal positions on the track. Timing and synchronization of multimedia items across different tracks are determined by vertical relationships of objects across tracks (similar to synchronization arcs). As well as media item tracks the Integrator allows input or control tracks, and a timing track which allows the timing of the tracks to be altered. Altering the timing is outside the scope of the temporal information specifiable within AHM, although could be added in a system-dependent manner to the temporal information in a temporal composite component.

The duration of a non-continuous item is derived from the track information, so, for example, an image will remain on display until another image occurs on the same track. Composite objects can also be created. A composite object can be placed on the control track of the timeline, and can be opened to view the layout of objects on its own timeline. Time dependencies between objects at different levels of the hierarchy cannot be specified because of the user interface.

Several "flow" operations can be added to the control track of the timeline, including iteration and conditionally branching constructs. Linking information could be extracted from these more general controls.

Transitions are attached to the media items rather than being associated with a link, and can occur at the beginning of an item or join two objects.

It is unclear whether a jump to a different part of the presentation can be specifi ed.

In AHM terms

An atomic component consists of a presentation specifi cation and content. The presentation specifi cation consists of a start time, a deduced duration, a channel reference and a transition style. A channel corresponds to an output device, possibly a window on a video display. It is unclear whether there are properties associated with a channel and whether other properties can be associated with the atomic component. The content is a reference to a media item (still images, video, audio). Anchor values can be specifi ed and can be associated with a link destination specifi ed as part of the information stored with the atomic component.

A temporal composite component is built up from atomic and composite components, in particular serial and parallel synchronization is possible. Synchronization arcs can be defi ned.

3.5.2.4 MET++

In the MET++ application framework [Acke94] a multimedia presentation is a hierarchy of serial and parallel compositions of media items. The building blocks consist of time layout objects and media objects, where each has a start time, a duration and an associated virtual timeline. These are incorporated into a hierarchical structure with the media objects as leaf nodes and the time layout objects as intermediate nodes.

The value of an attribute can vary over time, e.g. the horizontal or vertical position of an object.

In AHM terms

An atomic component consists of a presentation specifi cation and content. The presentation specifi cation consists of a start time, duration and spatial information, in particular position, which can vary over time. The content is a reference to a media item (2D and 3D graphics, images, text, audio, video and user interface components with event handling).

A temporal composite component is a hierarchy of atomic and composite components.

3.5.3 Hypermedia

3.5.3.1 Videobook

The Videobook system, [OgHK90], combines time-based, media composition with linking, allowing the construction of composite multimedia nodes among which a reader can navigate. A scene, consisting of media items and triggers, is played according to the timing and layout presentation parameters associated

with its children. Each scene can contain nested sub-scenes. Synchronization of objects is specifi ed by giving the start time of an object with respect to the scene. A trigger object, when selected by the reader, sends a message to its target object which is either displayed if it is a node or executed if it is a process.

In AHM terms
An atomic component consists of a presentation specifi cation and content. The presentation specifi cation consists of a duration, an extent and a position. The content is a media item (e.g. text, image, video, script). An anchor consists of an anchor value and a destination reference. An anchor's value is in terms of its extent, position and duration with respect to the atomic component. The destination of the implicit link is a temporal composite component or an atomic component.

Composition is temporal or atemporal. Temporal composition is a hierarchical collection of atomic and temporal composite components. Atemporal composition is a collection of temporal composite components (scenes, referenced by name).

3.5.3.2 Harmony
Harmony [FSMN91] integrates continuous media items into a hypertext system. Each object is considered a node and there are links between nodes. Links are used for expressing the timing relations between nodes. The notion of an object group is introduced, where, if an object group is the destination of a link, a message is broadcast to all members of the group when the link is traversed.

In AHM terms
Atomic component has content (text, music, graphics, video and animation) and associated procedures (in the object oriented sense). Anchors can be specifi ed in text, video and graphics media types.

A composite component is a temporal (parallel or serial) composition of atomic and composite components.

A link component is a separate component which also describes timing information.

3.5.3.3 HyTime
HyTime [ISO97b], [ISO97b] is a standard for representing the presentation independent structure of hypermedia documents. It embodies its own model of hypermedia, which includes complex hyperlinking, locating of document objects and the scheduling of objects within measured coordinate spaces such as space and time. As a meta-model, HyTime can be used to specify hypermedia models, such as the AHM. HyTime does not, however, provide constructs for presentation specifi c aspects of documents, which form an important part of the AHM. We have created extensions to HyTime architectural forms to express commonly found behaviour in hypermedia systems, in particular for modelling aspects of the runtime layer. These extensions comprise the Berlage[5] architec-

ture: shadow location form, aggregate link form, and duration marker form [ROHB97b].

In AHM terms
An atomic component and all its subparts can be expressed. Temporal and atemporal composite components and all their subparts can be expressed. Linking can be expressed. Channels can be expressed, although HyTime does not address this issue as directly.

The activation/deactivation cannot be expressed, hence the introduction of the Berlage aggregate link, which specifies that all the other children of a composite are deactivated whenever one of them is activated. This is a particular type of atemporal composition implemented in CMIFed, called the choice component.

Synchronization with objects of unknown duration cannot be expressed in HyTime directly, hence the introduction of the duration marker form.

3.5.3.4 MHEG-5
MHEG-5 [ISO97a], [JoRo95] was developed to allow a single representation of a multimedia presentation to be played on a range of end-user platforms over a distributed network. The presentation can be divided up into separate fully encapsulated parts which can be communicated separately, minimizing network traffic and the transfer of unnecessary information. MHEG-5 is defined as a collection of related object-oriented data structures and is a procedural language for which player software can be implemented.

In AHM terms
An atomic component has presentation specifications, anchors and content. The presentation specifications include start time and duration, a layout channel reference and style information. Anchors have an identifier, style information and a value. Content is a reference to a media item.

Temporal composition can be specified including presentation specifications and children. The presentation specifications include a duration, synchronization constraints and style. Atemporal composition is specifiable, in that non-temporally related presentations can be linked to.

A link specifier is implicit within the procedural specification of links. It includes a source context activation state with the choice of CONTINUE or REPLACE.

3.5.3.5 SMIL
SMIL, Synchronized Multimedia Integration Language, provides a declarative way of specifying multimedia documents for the World Wide Web. A specification of the language [BuRL91] and a high-level description [Bult97] are available. The work of the AHM and CMIF played a major role in the development of the

5. Berlage is an important Dutch architect.

requirements for the language. The SMIL requirements for simplicity of development of a player and readability of the syntax, however, have generally lead to simplifications of the model. One area where SMIL is broader than the AHM is the specification of alternative data formats for dealing with the delivery of the same document specification across differing network bandwidths.

In AHM terms
A channel has a position and an extent in terms of a browser window. An atomic component has presentation specifications and content. The presentation specifications include start time, duration and a channel reference and are able to refer to style information (using style sheets). Content is given by a reference to a URL. Atomic anchors can be specified or referred to in the content. Temporal composition is specifiable and is of two types: parallel and sequential. Synchronization arcs can be defined between direct children of a composite. A link has a source and a destination, where each has a component, an anchor and the source has a source context activation state. A source and destination anchor can be specified in terms of a reference to an anchor defined within the content of an atomic component, or in terms of a temporal/spatial anchor value, as described in Section 3.4.3.

3.5.4 Summary of system models expressed in AHM

AHM model elements			Hypertext				Multimedia				Hypermedia				
			Intermedia	**Guide**	**Microcosm**	**HTML**	**Athena Muse**	**Eventor**	**Integrator**	**MET^{++}**	**Videobook**	**Harmony**	**HyTime[a]**	**MHEG-5**	**SMIL**
Channel				y[b]		y[b]	y		y				y		y
	Pres. Spec.	Channel ref.											y		
		Position, extent		y		y	y		y				y		y
		Style		y		y									
	Attributes														
	Med. typ.						y						y		
Atomic Comp.			y	y	y	y	y	y	y	y	y	y	y	y	y
	Pres. Spec.	Duration					y	y	y	y	y		y	y	y
		Channel ref.					y						y		y
		Position, extent	?[c]					y[d]	y	y[d]	y		y	y	y
		Style	n[e]			y			y[f]					y	y
	Attributes				y	y			y[g]						
	Anchors		y	y	y[h]	y	y						y	y	
		Anchor ID	y	y		y							y		y
		Pres. Spec.				y								y	
		Attributes			y	y									
		Value	y	y	y	y	y		y		y	y	y	y	y
	Content	Media item ref.	y	y	y	y	y	y	y	y		y	y	y	y
		Data-dep. spec.											y		
Temporal Composite Comp.							y[i]		y	y	y	y	y	y	y
	Pres. Spec.	Duration								y			y	y	y
		Sync. arcs							y			y	y	y	y[j]
		Style												y	
	Attributes														
	Anchors												y		
		Anchor ID											y		
		Pres. Spec.													
		Attributes													
		List of anchors											y		
	Children	Comp. ref.					y		y	y		y	y	y	y

TABLE 3.8. Implicit document models of systems expressed in AHM

AHM model elements			Hypertext				Multimedia				Hypermedia				
			Intermedia	**Guide**	**Microcosm**	**HTML**	**Athena Muse**	**Eventor**	**Integrator**	**MET++**	**Videobook**	**Harmony**	**HyTime**[a]	**MHEG-5**	**SMIL**
Atemporal Composite Comp.			y[k]	y[l]	y[m]		y				y		y	y	
	Pres. Spec.	Init. activ. state		y										y	
		Play/pause													
		Style													
	Attributes														
	Anchors				y[n]								y		
		Anchor ID											y		
		Pres. Spec.													
		Attributes													
		List of anchors			y[o]								y		
	Children	Comp. ref.	y	y									y	y	
Link Comp.			y		y							y	y		y
	Pres. Spec.	Duration					y					y			
		Rel. position											y		
		Style	y												
	Attributes				y										
	Anchors		Links to links are not considered.												
		Anchor ID													
		Pres. Spec.													
		Attributes													
		Value													
	Specifi ers		y[p]	y	y	y							y		y
		Src. cont. activ.		y									y[q]	y	y
		Dst. cnt. pl./pa.	y												
		Style		y											
		Anchor	y	y	y	y	y	y[r]	y		y	y	y		y
		Context		y[s]		y[t]							y		
		Direction	y[u]		y[v]		y[w]	y[x]			y[w]		y		y

TABLE 3.8. Implicit document models of systems expressed in AHM

a. The table is fi lled in for HyTime as " can these elements be expressed directly using HyTime or SGML", and not as " does the HyTime model include these objects".
b. Areas can be defi ned within the main window
c. It is unclear whether this is recorded in the document or is decided at runtime.

d. The position can vary with time.
e. The anchor style is determined per media type rather than per atomic component.
f. In particular the transition special effect at the beginning or end of the display of the content.
g. In particular a user-specifiable name.
h. But part of link specifier, not of atomic component.
i. Temporal composition is one level only.
j. Only between direct children.
k. A temporal composition is of links, called a web.
l. An "include in text-flow" composition or a group composite.
m. A temporal composition is of links, called a linkbase, or of anchors, contained within the link.
n. Anchor composition is specified within a link specifier
o. Of the form: string specification in file "anywhere" at position "anywhere".
p. A link has two specifiers.
q. By means of the Berlage aggregate link form.
r. Where the anchor reference is whole component reference.
s. Source context is derived from the document structure.
t. The frame structure can be used to derive context.
u. Every specifier is BIDIRECT.
v. A link has one source and one destination specifier
w. The link component does not exist as such, but the relevant information can be extracted and stored as a link with two specifiers, one with direction FROM and the other TO.
x. It is not clear, but probable, that the implicit link has one FROM and one TO specifier

3.6 Conclusions

The Amsterdam Hypermedia Model has been designed to capture the elements of structure, timing, layout and interaction that are needed to specify an interactive, time-based on-line presentation. The AHM is not a perfect model of hypermedia, but instead seeks to achieve a balance of expressibility and implementability. In this chapter we have described the choices we have made, and motivated these choices. We have defined the model and shown that it can be used to describe the presentations created by a wide range of systems. We have shown that the model is sufficiently simple to be implementable by providing a description of the parts of the model which have been implemented in the CMIFed environment, Appendix 1.

The main components of the AHM are the channel, atomic component, temporal and atemporal composite components and the link component. Presentation specifications that can be associated with these components are selected from temporal, spatial, style and activation information.

Although AHM has its roots in the Dexter and CMIF models it incorporates the following novel extensions:

- The presentation specifications within the model have been explicitly stated as temporal, spatial, style and activation information. Each aspect occurs throughout the model and we have shown how the occurrences relate to one another.
- Anchors have been extended to include semantic attributes and presentation specifications, including start time and duration for an atomic anchor of a non-continuous media type.
- Content is specified explicitly as a media item reference along with a corresponding data-dependent specification.
- Anchor reference and channel reference in addition to a component reference are used throughout the model.
- Composition of anchors has been introduced.
- Composition of components is of two types: temporal and atemporal. Dexter expressed only atemporal and CMIF expressed temporal. Including both types of composition within one model requires the inclusion of activation state information.
- Activation state information has been incorporated throughout the model. This includes: initial activation state, play/pause state and change in activation state on following a link.
- Link components have been extended to include context in the link specifier
- Transition information, including transition duration and special effect, has been incorporated in the model.

The model could be extended in the following directions.

- Separate out the spatial layout hierarchy from the channel element and make a reference to it from a channel.
- Introduce spatial layout with respect to content.
- Include a way of selecting between synchronized streams of information, e.g. by introducing a visible/invisible state.
- Include a timeline more explicitly.
- Separate out link semantic direction from link traversal direction.
- Allow the specifi cation of anchors in terms of time and space for atomic and temporal composite components.
- Allow the inclusion of absolute time within the model. This could be associated with children of atemporal composites but would require an extension to the current link activation mechanism.
- Include auto-fi ring of links.

A formal description of the model in the Object Z language is given in [OsEl97], included as Appendix 2 of this thesis.

The following chapters investigate the authoring aspects of hypermedia presentations.

4 Multimedia Authoring Paradigms

The construction of a coherent hypermedia presentation composed from its constituent parts is a non-trivial task. To explore the requirements of a hypermedia authoring system designed to aid an author in this task, we describe a selection of both research and commercial authoring systems. These provide examples of the types of support that can be given to authors, and how this support can be provided in practice. We differentiate four authoring paradigms and discuss their advantages and disadvantages for editing features of a multimedia document model.
This chapter is based on work presented in [HaBu95b].

4.1 Introduction

In the previous two chapters we discussed and proposed a hypermedia document model which can be used to record sufficient information for storing a hypermedia presentation. A good model is a necessity for a good tool. A model only, however, is insufficient for solving the problem of how to create such a presentation. Tools are required for the creation, manipulation and deletion for individual parts of the model but in addition to such basic requirements the author needs an environment which supports the complete authoring process.

This and the following two chapters discuss authoring environments for multimedia and hypermedia documents. This chapter illustrates a selection of approaches implemented in existing systems. The following chapter, Chapter 5, states the full requirements for a hypermedia authoring environment and Chapter 6 describes the editing environment CMIFed.

We begin our analysis of the requirements for a hypermedia authoring environment by investigating a number of multimedia authoring systems that exist as either academic prototypes or as popular commercial systems. Each of these systems allows the creation of a multimedia presentation conforming to the system's own proprietary document format and uses its own suite of tools for creating the presentation.

Constructing a presentation consists of three major processes:

- creating and editing the media items comprising the presentation;

- assembling the items into a coherent presentation, where this includes the specification of the temporal and spatial layout of the items; and
- specifying the interaction between the reader and the presentation.

Our analysis of the authoring task concentrates primarily on designing an authoring system that supports the last two of these—the assembly and interaction processes. We are less concerned with the first—the creation of individual media items—which requires the use of specialist data editors for the range of media types used. Neither do we focus on the data formats used, nor the necessary trade-offs for authoring presentations destined to be played over a network.

In some respects, authoring multimedia can be compared with word processing. Both activities require the collection/generation of source material and the placement of these sources within a presentation environment. A generic word processor 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, in the document. In the same way, multimedia authoring tools allow an author to integrate several types of information into a composite presentation. Unlike text, however, the temporal dimension often dominates the multimedia authoring process. In many respects, then, multimedia authoring is more akin to movie making. Here an editor is concerned that the individual shots that have been created are assembled into sequences which are in turn are grouped into scenes containing a single coherent thread of the story [RuDa89].

An author of multimedia has the same goal of communicating a message to the reader. In order to achieve this goal, the author is required to specify the individual parts of a multimedia document. To ease the task for the author, these specifications should be as transparent as possible and retain the emphasis on the manipulation of the message rather than on the document parts. This requires the presentation of the document parts to the author in a way that supports higher-level narrative manipulation. We term the different approaches used for this authoring paradigms. An *authoring paradigm* presents the author with a particular view of the document model. For example, in the word processing example a document can be viewed as a sequence of words in a text flow or as a layout of areas on a page. In the movie world the paradigm is the grouping of shots into sequences and scenes.

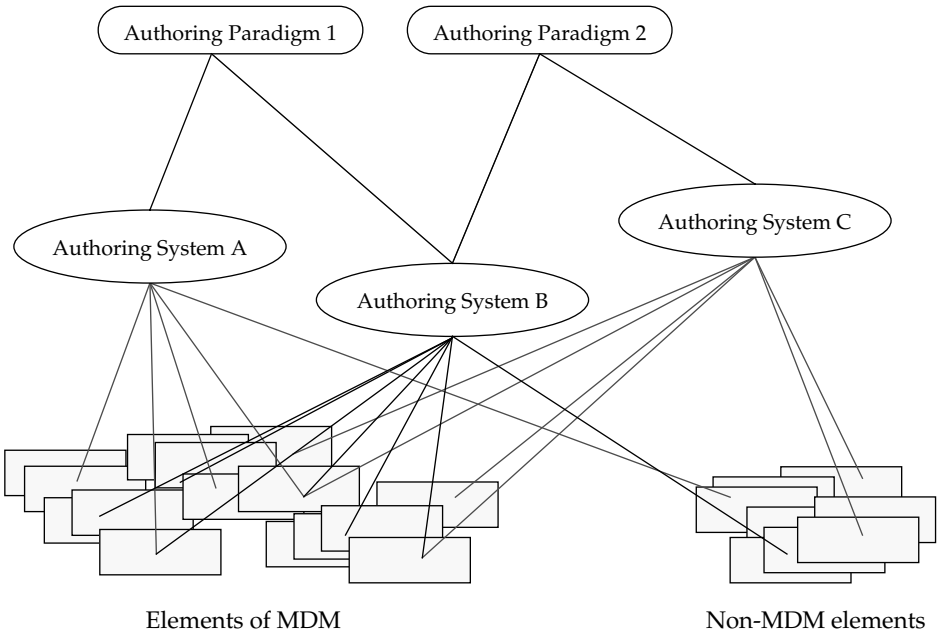
In this chapter we analyse the authoring approaches and individual functionalities of existing multimedia authoring systems. In order to compare these, however, we discuss them in terms of a multimedia document model. This model is less rich than the Amsterdam Hypermedia Model, defined in the previous chapter, since most multimedia systems do not, for example, include explicit link objects. Also, where objects that do correspond to the AHM are explicit they

tend to have a less complex structure, e.g. composition of instances. The multimedia document model is not all encompassing, so some authoring systems may manipulate objects that lie outside the scope of the model, e.g. a user interaction history.

While the document model that is manipulated by these systems is similar, there are a number of distinct authoring paradigms which are used. An impression of the relationships among authoring paradigms, authoring systems and the parts of a document model is given in Fig. 4.1.

Having described a selection of authoring systems, illustrative of the authoring paradigms, we make an analysis of which paradigms are more supportive for editing which parts of the document model.

This chapter is structured as follows. We first give two sets of definitions: the authoring paradigms used for categorising authoring systems; and a multimedia document model used for comparing authoring facilities. In Section 4.3 we use these definitions to discuss a representative selection of academic and com-



An authoring system is able to edit elements of a multimedia document model. The approach used may conform to a single authoring paradigm, but is more often some combination of paradigms.

Figure 4.1. Relationship between authoring paradigms, authoring systems and elements of a multimedia document model (MDM).

mercial multimedia authoring systems. We use this as a basis for analysing the authoring paradigms for their suitability for the authoring tasks. We conclude with a summary of our analysis.

4.2 Definitions

In this section we present terminology for classifying multimedia authoring systems and discussing their features. We first present a multimedia document model to allow the discussion of individual features of the systems in terms of the document structures they manipulate. We then present four authoring paradigms which are used to classify authoring systems in terms of the style of interaction provided to the author.

4.2.1 Multimedia Document Model

To serve as a base for discussing multimedia authoring systems, we define elements of a multimedia document model. The model, which is in some respects similar to the AHM (defined in Chapter 3), is not dependent on any higher order information structuring. Fig. 4.2 gives an overview of the multimedia document model.

- A *media item* is the data associated with a single playable object in a multimedia presentation, for example a piece of text, an image, a video or a sound fragment. It may also be a piece of program code or a combined video/audio format.

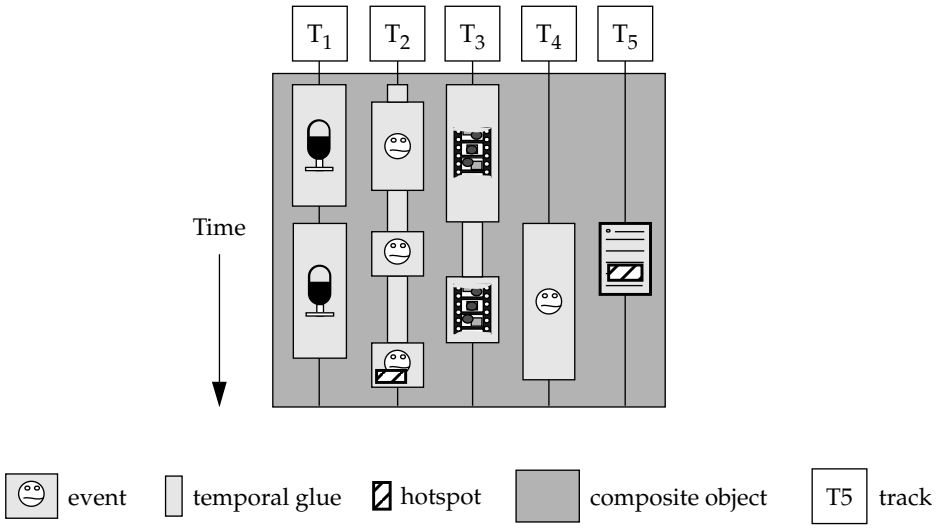


Figure 4.2. Multimedia document model overview

- An *event* is an action that occurs during a multimedia presentation. It may include a reference to a media item, e.g. play the media item for a specified duration, or it may contain more control-oriented information, such as wait one minute then jump to another part of the presentation. An event is an action and a media item is an object. Where only objects are specified in the multimedia document, it is up to a document player to interpret what to do with the objects, i.e. to make the translation from object to event.
- *A composite object* is used to refer to a collection of media items. While it is similar to an AHM composite component it does not possess properties beyond the list of its children. In other words it has no associated presentation specification, attributes or anchors specific to the composite.
- *Tracks* allow media items, events, or control information to be collected together in a single stream.
- *Constraints* are specifications of temporal or spatial relationships between uses of media items. An example of a temporal constraint is the AHM synchronization arc, e.g. start displaying an image 2 seconds after a piece of music begins. An example of a spatial constraint is that a text label should be placed centred at the bottom of an image.
- *Temporal glue* has a duration but no associated media item. This can be captured as an AHM atomic component that has a duration but no associated content.
- A *transition* is a presentation effect used when the system finishes displaying one media item and starts displaying another, e.g. a video item dissolves to the next video item.
- *Hotspot* or *button*. Most multimedia authoring systems have no explicit structures for anchors and links. They often, however, allow the specification of something that corresponds to an anchor value, e.g. an area of an image or a text string. This can be visualised, for example by drawing a border around it or using a different colour, and the reader can click on it. It is this visualisation, rather than the underlying structure, which is referred to as the hotspot or button.

4.2.2 Authoring Paradigms

The majority of multimedia authoring systems can be classified according to a number of different underlying paradigms: *structure*, *timeline*, *flowchart* and *script*. The paradigms provide different approaches to authoring. While we use these to classify the authoring systems discussed in the next section, more than one paradigm may be present in any one system.

Structure-based

Structure-based authoring systems, Fig. 4.3, support the explicit representation and manipulation of the structure of a presentation. The structure groups media items included in the presentation into "sub-presentations" which can be manip-

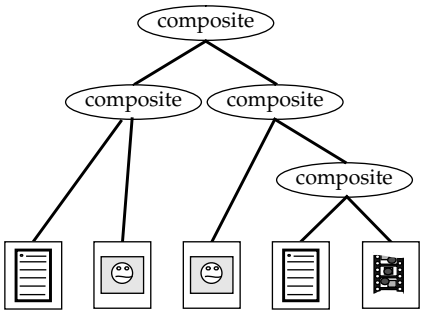


Figure 4.3. Structure-based paradigm

ulated as one entity, and thus can in turn be grouped. Although in principle the same object can belong to one or more groups, in current authoring systems this is not the case. The destinations of choice points in a presentation, that is where the reader is able to select to go to other parts of the presentation, are given in terms of the structure. The structuring may group the media items indirectly, where, for example, higher-level concepts are associated with each other and each concept is associated with one or more (groups of) media items.

Timeline-based

Timelines show the constituent media items placed along a time axis, possibly on different tracks, Fig. 4.4. These give an overview of which objects are playing when during the presentation. Timeline based authoring systems allow the specifi cation of the beginning and end times of display of a media item in relation to a time axis. Manipulation is of individual objects, rather than of groups of objects, so that if the start time or duration of a media item is changed then this change is made independently of any other objects placed on the timeline. The destinations of choice points are given in terms of a new position on the timeline.

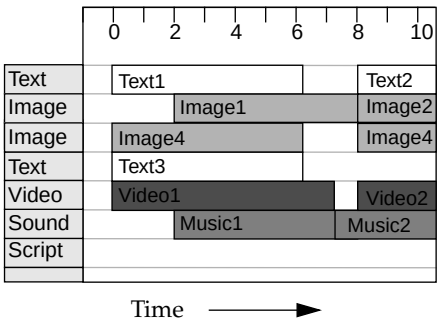


Figure 4.4. Timeline paradigm

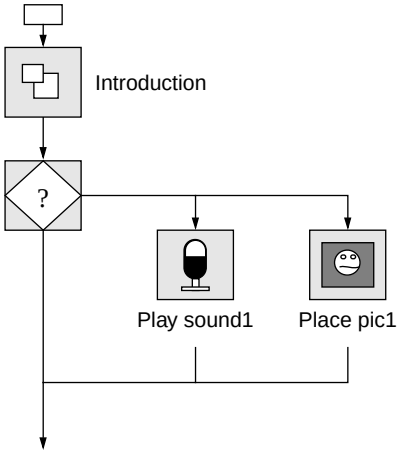


Figure 4.5. Flowchart paradigm

Flowchart
A flowchart gives the author a visual representation of the commands describing a presentation, Fig. 4.5. Authoring with a flowchart is similar to programming the presentation in a procedural way, but with an interface improved by icons for visualising the actions that take place. The narrative of the presentation can be reflected in the routines and subroutines used. The order of displaying or removing objects and other events is shown, but time is not represented explicitly. The destinations of choice points are given in terms of jumping to a new procedure.

Script-based
A script-based system provides the author with a programming language where positions and timings of individual media items, and other events, can be specifi ed, Fig.4.6. Authoring the presentation *is* programming. The destinations of choice points are given in terms of jumping to a new procedure.

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

Figure 4.6. Script-based paradigm

Discussion
Each of these paradigm described is based on a particular view of a multimedia presentation. The structure-based paradigm emphasizes the narrative structure of the presentation; the timeline emphasizes the temporal aspects; the flowchart and script emphasize the execution order of displaying and removing objects at runtime. The structure and timeline paradigms are of a more declarative nature, while the flowchart and script paradigms are procedural. The first two specify structural or timing properties of objects which are then interpreted as events by a system at play-back time. The second two list a sequence of actions, including those referring to media items, which are executed by the system.

While all four authoring paradigms include the notion of event, it is more explicit in the flowchart and script based paradigms. The flowchart and script paradigms are based on streams of events, where either an icon or a script statement specify what the event is. The structure-based paradigm manipulates composite and media items which are interpreted by the system at play-time as events. A timeline refers to objects representing media items, but, given that their presence on the timeline implies that they be played for the specified duration, are more correctly events. Other events, such as increasing the tempo of the presentation, can be difficult to visualize on the timeline.

The paradigms themselves are not mutually exclusive, but reflect a difference in emphasis. It is not that any one approach provides the ideal solution to an author's task and more often a combination is appropriate.

4.3 Analysis of multimedia authoring paradigms

The goal of this chapter is to compare the advantages and disadvantages of existing authoring paradigms. This section describes a number of authoring tools illustrating each of the four paradigms discussed in the previous section. The descriptions provide a basis upon which a comparison of different paradigms can be made. Most of the authoring system described are to be found in the academic literature, since they explore the more innovative approaches. We have also chosen to include a number of commercial systems, since these have proven themselves by surviving years of use by real authors. While a large number of authoring systems is commercially available we have selected Director, Authorware, and IconAuthor as being representative of the commercial systems.

Each sub-section describes a number of systems in terms of the paradigms and terminology discussed in the previous section. While a number of the systems described in this section use more than one of the paradigms described above, we have used the predominant paradigm to classify the system. The paradigms should be viewed as descriptive of the approach(es) taken by a system, and are used as a basis of making comparisons among systems. We wish to emphasize the strengths and weaknesses of the paradigms, rather than recommending one

authoring system above another. For each paradigm we have selected one system which we consider as typifying the paradigm. Where other systems illustrate extra features or insights we describe these also. The descriptions of the systems highlight authoring features that will be referred to in the discussion of desired features. This section is not a review of the best authoring system to purchase.

4.3.1 Structure-Based Authoring Systems

While our own system CMIFed is a structure-based authoring system, we discuss it in detail in Chapter 6 and omit it here.

4.3.1.1 MAD

MAD (Movie Authoring and Design) [BRFS96] decomposes a multimedia presentation as a nested hierarchy, Fig. 4.7. This hierarchy is able to represent "acts", "scenes" and "shots", although these divisions are not imposed on the author. In a manner similar to a text outliner, the different levels of the hierarchy can be hidden or revealed and the position of items can be moved within the hierarchy. The start time of each item is calculated from the start times and durations of preceeding items and subitems in the hierarchy. The duration of an item can be calculated on the basis of the media item for video and audio, or can be specified by the author. The author also has control over playback of sections of the presentation by playing complete items or skipping forward to following items. MAD lacks any control of synchronization among items so that a single item with its associated parts can be played, but other items cannot start before it has finished. It is unclear to what extent there is control of spatial layout.

4.3.1.2 MET++

In the MET++ authoring system [Acke94] a multimedia presentation is considered to be a hierarchy of serial and parallel compositions of media items. The temporal layout of the constituent items is derived from this composition hierarchy automatically. The building blocks consist of composite objects, called time layout objects, and media objects. Each has a starting time, a duration and an associated virtual timeline. The media object contains a reference to a media item and associated attributes, such as position, which can vary with time. Both object types are incorporated into a hierarchical structure with the media objects as leaf nodes and the time layout objects as intermediate nodes. When the start time or duration of an event is altered all time positions are recalculated. Any object can be stretched or reduced in time. In the case of a composite object, the transformation is applied throughout the descendant hierarchy.

The timeline representation, Fig. 4.8, allows the visualization and manipulation of the hierarchical structure–combining structure and timing information in one representation. The timeline shows the values of attributes that vary over time, e.g. the horizontal and vertical positions in the figure. This representation

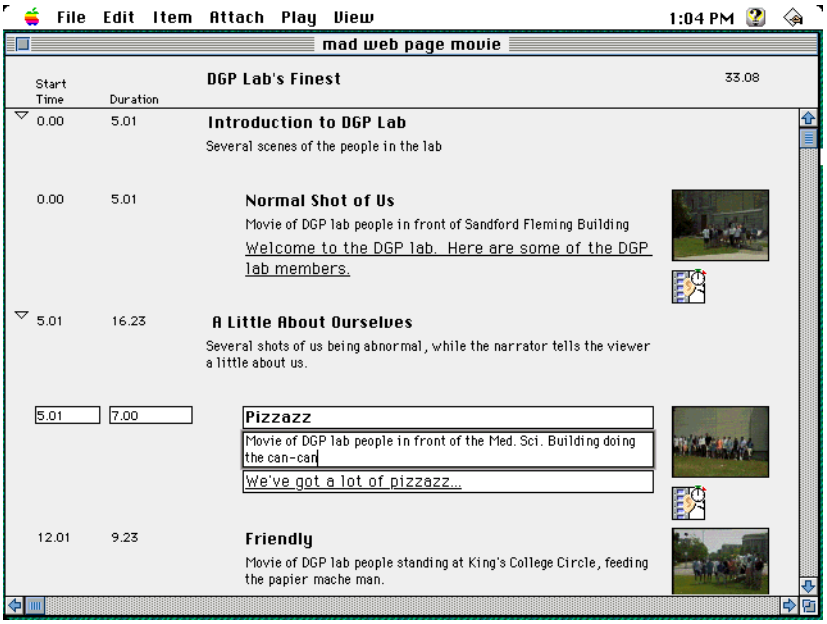
could also be used for other object parameters, e.g. the volume of an audio object, or the fade-in rate of an image or video.

4.3.1.3 Mbuild

The multimedia authoring system Mbuild [HaRe94], Fig. 4.9, uses a hierarchical structure of composites and multimedia items for editing and reusing composite multimedia data. The timing of the presentation is determined when the highest ranking composite object is determined and is calculated using temporal glue. Authors are able to create empty hierarchical structures, reflecting the narrative of the presentation, and only later fill them with the desired media items.

4.3.1.4 Discussion

Structure-based systems allow the explicit specification and manipulation of a presentation's structure. The advantage of this is that authors are able to use the structure as a storyboard, i.e. a representation of the narrative, for the presentation. The author is thus able to manipulate the narrative directly. Since the presentation consists of different levels of structure, this can be viewed at different



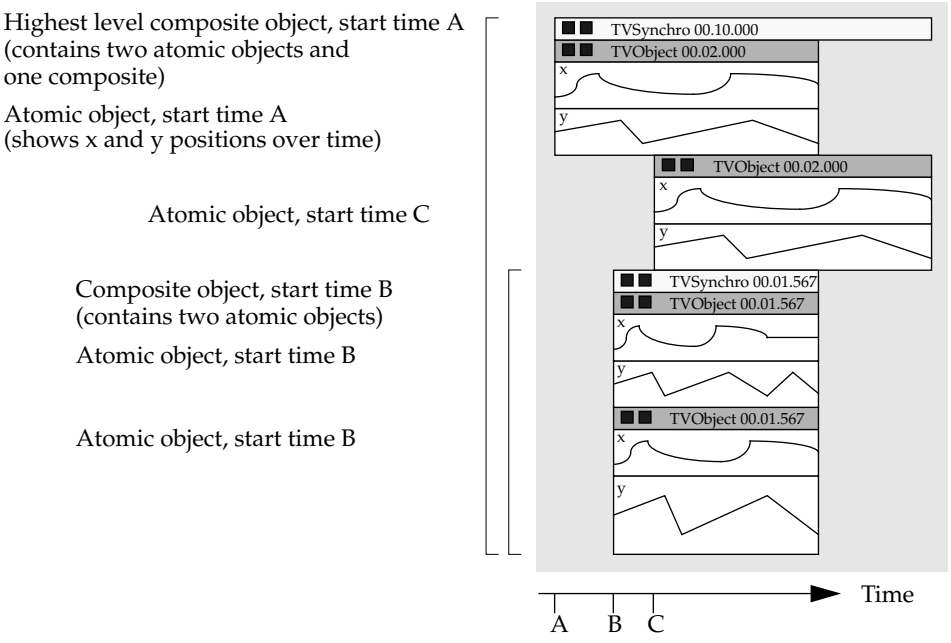
The indentation shows the level in the hierarchy. Each item in the presentation has three fields: title (bold), screen directions (small) and narration or dialogue (underlined) and may also have associated commentary, music, video or storyboard frames—shown to the right of the script. Start times and durations are shown to the left of the script.

Figure 4.7. MAD (Movie Authoring and Design) script view

levels of detail, allowing the author easy navigation of the narrative. Another advantage is that since the structure is able to indicate an ordering it can be used for deriving the timing for the presentation, as demonstrated in MAD, MET++ and Mbuild. The timing can thus be visualized and edited, at least to some extent, in the structure-based view. It may even be possible to have the structure displayed along a timeline, as illustrated by MET++.

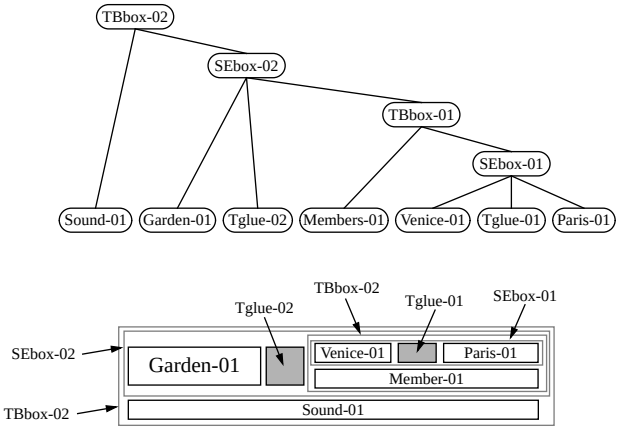
Synchronization constraints can, at least in principle, be defined between media items, between a media item and a scene (or other structure), or between two structures. While the fact that a timing relation exists could be shown in a structure-based view, it requires a time-based view to show the actual influence of the constraints specified. One example is the Firefly system, [BuZe93], where timing constraints, of some complexity, can be defined, but since the view is not time-based the resultant timing is not visualized.

Spatial layout is defined by assigning a position on the screen to the media item. This can be done by positioning the item where it should appear or, as in



Each *TVObject* represents a media item and specifies its *x* and *y* positions. *TSynchro* titles are composites of the objects below them. Both types of objects can be cut, copied, and pasted, stretched and shrunk. (Adapted from Fig. 7, [Acke94].)

Figure 4.8. Time composition view in MET++.



The upper part of the figure shows the hierarchical structure of a presentation. The lower part shows the equivalent structure as displayed in *Mbuild*. (Adapted from Fig. 20, [HaRe94].)

Figure 4.9. Hierarchical structure in *Mbuild.*

MET^{++}, by specifying the x and y positions over time. Neither method is specific to the structure-based paradigm, and in both cases it is difficult to get an overview of the position of the object relative to other objects or over time.

Links can be created among structures, allowing, at least in principle, source and destination contexts to be specified along with the source anchor (i.e. the value from which the hotspot is derived).

A problem with the structure-based paradigm is that extra authoring effort has to be expended to create the initial structure. Our hypothesis is that the benefits of understanding and manipulating of the presentation's structure will outweigh the initial effort.

A purely structural view of the presentation gives no understanding of the timing of the presentation. This can, however, be combined with the timing information, as demonstrated in MET^{++}, Fig. 4.8, and Mbuild, Fig. 4.9. Where structural and timing information cannot be combined, multiple views of the presentation can be a solution.

A summary of the properties of the structure-based paradigm is given in Table 4.1.

4.3.2 Timeline-Based Authoring Systems

4.3.2.1 Director

Director [Macr97] is a commercial system designed for creating animation-based presentations. Graphics, text, audio and video media items can be placed on a timeline, or score as it is termed in the system, Fig. 4.10. The timeline is divided

into discrete time intervals, called frames, whose speed of playing is determined by the current rate of play, called tempo. The tempo can be changed at any frame. The timeline has a number of associated tracks, where, apart from a number of effects tracks, any media item can be placed on any track. A media item has a position in each frame, and the author can describe a path for the media item to follow through a series of frames. Sections of the timeline can be cut, copied and pasted. Jumps to other parts of the timeline are implemented via a " goto frame"command in the scripting language. Each frame, media item or anchor within a media item, can have an associated script. The script is executed when the end-user interacts with its associated object, normally by clicking with a mouse.

Scene breaks can be recognised by the author by sudden changes of media items on the timeline. These are not automatically marked by the system since there are no special frame types (e.g. a " beginning of scene"frame), nor groupings of frames. An author is, however, able to add explicit markers to frames, allowing jumps to the marked frame, and thus to the marked beginning of a scene.

One of the effects tracks is a transition track, allowing the specifi cation of transitions. The transition is recorded with the fi rst frame of the following sequence, rather than the last frame of the previous one. The transition has a type (for example dissolve or checkerboard), a duration and a choice of whether the whole display area is affected or only the differences between the frames.

4.3.2.2 The Integrator

An explicit goal of the authors [SFHS91] is to create a development environment for interactive multimedia that relieves a developer of programming work. The central tool in the environment is the high-level Integrator which is used to assemble various media items into a multimedia application and to specify ways of interacting with the application.

Property / Paradigm	show narrative structure	edit narrative structure	navigate narrative	show timing	edit timing	show synchronization	edit synchronization w.r.t. objects	edit synchronization w.r.t. structure	show layout	edit layout	show interaction	edit interaction	interaction diversity
Structure-based	++	++	++	0	+	0	+	++	--	0	-	0	-

Key: ++ very good, + good, 0 neutral, - bad, -- very bad/not possible

TABLE 4.1. Properties of structure-based authoring paradigm

The basic paradigm used in the Integrator is the timeline, where a pool of media items can be sequenced in virtual time, and then mapped for display onto real time. In the Integrator the multimedia presentation is represented as a set of tracks. Timing and synchronization of multimedia items within a single track are determined by their horizontal positions on that track. Timing and synchronization of multimedia items across different tracks are determined by vertical relationships of objects across tracks (similar to AHM synchronization arcs). As well as media item tracks, the Integrator allows input or control tracks and a timing track which allows the tempo of the tracks to be altered, similar to Director.

Authoring is carried out by placing an icon representing a media item on one of the tracks at a specifi c time. A static item, such as an image, will remain on display until another item occurs on the same track.

While the main authoring metaphor is the timeline, composite objects can also be created, e.g. an image with a graphic overlay, or a slide show. A composite object can be placed on the control track of the timeline, and can be opened to view the layout of objects on its own timeline. The composite object appears as one object on the timeline, which makes it diffi cult to get an overview of all the

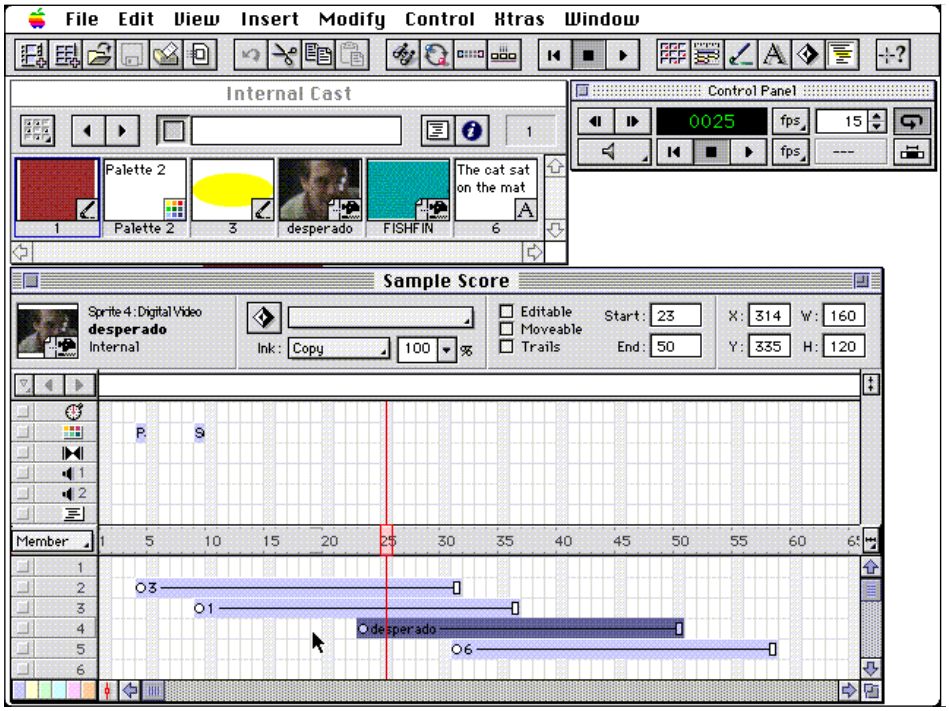


Figure 4.10. Director

objects making up the presentation. Also, time dependencies between objects at different levels of the hierarchy are impossible to specify.

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

Transitions can be specified in the system. These are associated with the media item events rather than being separate events themselves, and can occur at the beginning of an event (e.g. fade up from black), at the end of an event, or can join two events (e.g. a video dissolves to another video).

4.3.2.3 Discussion

Timeline-based authoring centres around presentation over time. It uses a time-axis as the main method of organising the (temporal) positioning of media items in the presentation. It is visualized as a line with marked-off time intervals. The advantage of this approach is that the start times and durations of the media items in the presentation are displayed explicitly, and in principle can also be manipulated directly. The timeline can also be used to show the values of properties of the media items that vary over time (as demonstrated in MET^{++}).

A further advantage of the timeline is that synchronization constraints, where these exist, can also be shown and in principle manipulated directly. These constraints can be between media items or between a media item and the timeline. We, however, feel that the synchronization conditions should be expressed directly between (parts of) the media items themselves, so that if other durations are changed the system, rather than the author, can resolve the specified constraints. For example, if a video sequence is shortened or lengthened the corresponding subtitles stay synchronized with the correct parts of the video.

Spatial layout is specified by assigning a position on the screen to the media item. This can be done by positioning the item where it should appear or by specifying the x and y positions over time. Neither method is specific to the timeline-based paradigm. It is difficult to get an overview of the position of the object compared with other objects and over time. Although no overview is available, the timeline does provide the author with easy access to a screen view from any point along the timeline.

Where links are specified, these are via scripts associated with an object being displayed on the screen. The script defines the destination of the link and may also contain some sort of transition information, such as "dissolve to next scene in 2 seconds".

The main problem with only a time-based representation is that for long presentations it is difficult to navigate around. Also scene breaks, or any overview of the narrative, need to be recognised implicitly, for example, by an abrupt change

in the objects on the timeline. Because scenes are not represented explicitly it is also not possible to create synchronization constraints in relation to a scene. Control flow can be added to a presentation as an object on the timeline but its effect cannot be visualized using the timeline.

A summary of the properties of the timeline-based paradigm is given in Table 4.2.

4.3.3 Flowchart-Based Authoring Systems

4.3.3.1 Authorware

Authorware (chapter 12 of [Bufo94], [BuHe93][1], [Macr97]) is a commercial system for creating interactive multimedia presentations for computer based training and kiosk applications, Fig. 4.11. To create a presentation, icons representing actions are selected and incorporated into a flowchart defining the sequence of events in the presentation. Flowcharts can be grouped into subroutines and nested to arbitrary levels. This is often necessary, since there is a limit to the display area for any one flowchart. The hierarchy of subroutines can be used by the author as an outline, or storyboard, for working on the presentation top down—first by stating the sections in the presentation and then filling them in. The flowcharts remain procedural however, and there is no way of getting an overview (via a timeline) of which media items will be played on the screen when. Interactions, on the other hand, can be fairly complex and go far beyond links, which are implemented as "jump to the" commands.

Property / Paradigm	show narrative structure	edit narrative structure	navigate narrative	show timing	edit timing	show synchronization	edit synchronization w.r.t. objects	edit synchronization w.r.t. structure	show layout	edit layout	show interaction	edit interaction	interaction diversity
Timeline	-	--	-	++	+	++	++	--	+	0	--	--	-

Key: ++ very good, + good, 0 neutral, - bad, -- very bad/not possible

TABLE 4.2. Properties of timeline-based authoring paradigm

1. The packages Authorware and IconAuthor were compared in [BuHe93] in 1993. Note, however, that both packages have newer versions on the market which may differ substantially from those reviewed.

4.3.3.2 IconAuthor

IconAuthor ([Aimt97], chapter 12 of [Bufo94], [BuHe93][1]) is a commercial package providing a suite of editors for different data types. Objects created by these, and other external editors, can be assembled in the central application builder for inclusion in the presentation. This is icon-based with flowcharts constructed from a library of icons representing actions comparable to those found in conventional programming languages and other more specialist icons for media-presentation and interaction. There is no enforcement of any programming discipline, so that large, unstructured graphs can be created. The author is given some help with flowchart navigation through being able to zoom in and out of the flowchart representation, and being able to simplify the display by collapsing or expanding collections of icons. Previewing the presentation is possible from the beginning or from a selected starting point.

4.3.3.3 Eventor

The creators of Eventor (Event Editor), [ENKY94], argue that authoring facilities should apply a divide and conquer approach, which they support by providing three different views of the presentation–temporal synchronizer, spatial synchronizer and user interaction builder. They distinguish timeline-based and

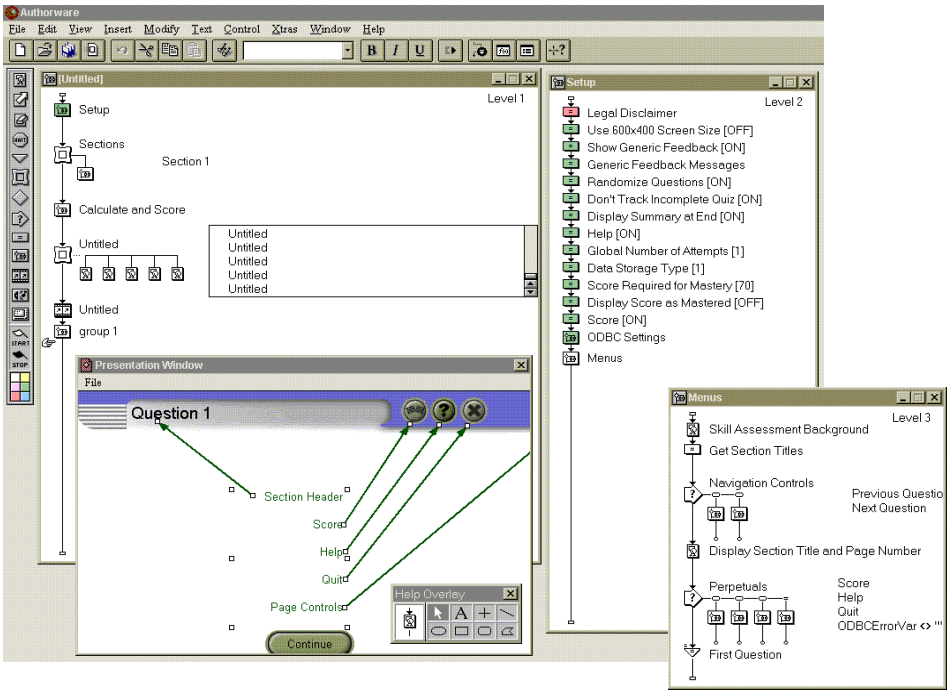


Figure 4.11. Authorware

flowchart paradigms (which they refer to as event-based), and aim to incorporate the advantages of both in an authoring system. Eventor is based on CCS, Calculus of Communicating Systems, a formal specification mechanism. This allows a formal specification of the behaviour of the presentation, which can be used for checking, for example, syntactic correctness. In addition, they provide automatic aids for validating, for example, temporal constraints.

Basic units of programming in Eventor are media items (called basic objects). Temporal synchronization can be specified among media items and composite objects. Synchronization points (similar to AHM anchor values) can be marked in video. The temporal synchronizer visualises the composite objects in a flowchart style structure. The spatial synchronizer allows the author to specify paths and scaling transformations by demonstration—these are tightly coupled to the temporal synchronizations. (While this provides a more direct method of interaction than in MET++, section 4.3.1.2, the latter visualises the spatial movements in time more explicitly, Fig. 4.8.)

4.3.3.4 Discussion

In a flowchart, control is the emphasized view: events are executed in turn, determined by the surrounding control structure. The advantage of the flowchart paradigm is that it incorporates more powerful interaction commands. For example, standard multiple choice question formats are often provided which support the creation of links to a different section from each answer.

The paradigm provides some form of abstraction, but this is a grouping of commands in the form of nested flowcharts rather than relations among (groups of) events. This allows the narrative structure of the presentation to be reflected by the different levels of flowcharts. Although this is a useful view of the presentation's structure, it is difficult to find which items are displayed on the screen in the middle of a flowchart, since, e.g., background images may have been displayed before the flowchart was executed. This means that a sub-scene cannot be played independently—since the state of the presentation is known only by playing the presentation.[2]

The only form of timing specification is through the use of "display item" and "erase item" commands in the flowchart. This leads to three disadvantages. Firstly, if a number of items are to be displayed simultaneously then this cannot be specified, but only approximated by using a number of "display item" commands one after the other. Secondly, synchronization relations among items cannot be specified. Thirdly, it is not clear from the script which objects are on display at any particular time. This could, however, in principle be calculated from the scripts and displayed in a different view (as in, e.g., Eventor). A timing overview is sorely needed in this paradigm, since an object may be displayed at

2. Authorware provides a choice of playing a single item or playing from a prespecified flag. IconAuthor plays from a selected starting point.

Paradigm \ Property	show narrative structure	edit narrative structure	navigate narrative	show timing	edit timing	show synchronization	edit synchronization w.r.t. objects	edit synchronization w.r.t. structure	show layout	edit layout	show interaction	edit interaction	interaction diversity
Flowchart	+	0	+	-	--	--	--	--	--	-	-	+	+

Key: ++ very good, + good, 0 neutral, - bad, -- very bad/not possible

TABLE 4.3. Properties of fl owchart-based authoring paradigm

the beginning of a long script and erased only at the end. For the same reason, an author may forget to erase an object when it is no longer needed.

Just as timing is given through the use of commands, so is the spatial information for an object. Where the other objects are placed on the screen is only to be found by looking through the fl owchart. Any overview of the objects in time, or their relations with respect to other objects is thus not available. The Eventor system introduced the spatial synchronizer to help with this problem.

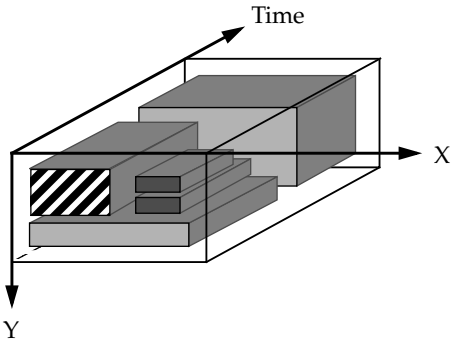
Links are specifi ed via commands associated with hotspots which defi ne which playing objects should be erased and which new objects should be displayed.

A summary of the properties of the fl owchart-based paradigm is given in Table 4.3.

4.3.4 Script-Based Authoring Systems

4.3.4.1 Videobook

The Videobook system, [OgHK90], was designed to incorporate a time-based, media-composite data sequence with the hypertext node and link concept, allowing the construction of composite multimedia nodes. The system presents media items and hotspots according to a script specifying their presentation parameters—timing and layout. The script is visualised as a thr ee-dimensional display showing the layout of each object along a timeline (Fig. 4.12). The system thus provides a low-level scripting language for the author to specify a presentation, which is then given a higher-level visualization along a timeline. The author is provided with some amount of structuring support, since each scene is defi ned in a separate script and scripts for scenes can contain nested sub-scenes. Synchronization of events is specifi ed by giving the start time of an event with respect to the scene. Although the multimedia document has an underlying structure-based paradigm, the structure is interpreted from the author-defi ned



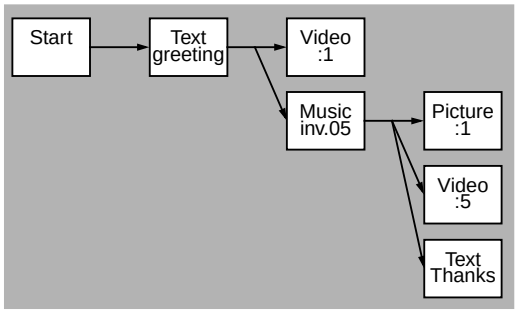
The layout and start times of the media items, defi ned by the author in a script, are given a three-dimensional visualisation.

Figure 4.12. *Videobook* scene.

script, rather than being used as the basis of editing and for generating the script.

4.3.4.2 Harmony

Harmony [FSMN91] has goals similar to those of Videobook, integrating dynamic media items into a hypertext system. Each object is considered as a node and there are links between nodes. Links are used for specifying the timing relations between nodes, using expressions such as `<aVideo, started: 30, aMusic, play>` which specifi es that the piece of music starts 30 seconds after the video fi nishes. The notion of a composite object, or object group, is introduced, where, if a composite is the destination of a link, a message is broadcast to all members of the composite when the link is traversed. A scenario viewer displays the (derived) structure of a scenario. This is illustrated in Fig. 4.13, showing the time ordering or relations among media items.



The presentation structure as shown in the Harmony scenario viewer.

Figure 4.13. *Harmony* scenario structure.

4.3.4.3 Command Streams
Some authors have taken the script form of specification even further[HeKo95]. Here, the authors view a presentation as a sequence of (possibly synchronized) command streams, where each stream consists of an ordered collection of commands, each of which is assigned its own execution time. If a command stream starts to fall behind, then commands can be skipped to allow the stream to catch-up. The command stream contains sufficient information to allow it to be played not only backwards, but also in both directions at a higher speed (to allow the presentation to be scanned until the reader finds the appropriate part).

It is interesting to note that deficiencies the authors plan to resolve in a revised model are grouping the commands into logical groups (i.e. the introduction of hierarchical structure) and knowing the duration of an object (derived from its first appearance on the screen and its later removal) so that it can be skipped when large jumps in the presentation are made.

4.3.4.4 Discussion
The script-based systems are in essence similar to the flowchart in terms of their flexibility and power of expression, but through the direct use of the scripting language are likely to be more flexible.

In terms of authoring support, however, they lack tools for viewing the procedure calls in any structured way. This in turn leads to more likely program structure errors. Even if the narrative structure of the presentation has been reflected in the script structure, then it remains difficult to manipulate at a high level. As with the flowchart, timing information for the presentation is embedded in the lines of code (with the exception of command streams [HeKo95] where the lines of code have explicit times). Spatial layout information is also given via the lines of code. Since the structure, timing and spatial layout information is present in the code it is possible to derive a structure or time-based visualisation. For example, in Videobook the space and time coordinates of items are shown in a 3D time-based representation and in Harmony the structure can be viewed.

With no further support, this is a tedious, low-level method for specifying a multimedia presentation. It can, however, be the most flexible, since with a more general language the author is not restricted to the actions supplied by an authoring system for manipulating a document model.

A summary of the properties of the script-based paradigm is given in Table 4.4.

4.4 Conclusions

Events and Links
All systems described in this section are in principle capable of creating similar presentations, through specifying events and links. A complete event consists of a media item that can be played as part of the presentation, its (possibly derived)

Paradigm \ Property	show narrative structure	edit narrative structure	navigate narrative	show timing	edit timing	show synchronization	edit synchronization w.r.t. objects	edit synchronization w.r.t. structure	show layout	edit layout	show interaction	edit interaction	interaction diversity
Script	-	-	-	--	--	--	--	--	--	-	-	0	++

Key: ++ very good, + good, 0 neutral, - bad, -- very bad/not possible

TABLE 4.4. Properties of script-based authoring paradigm

start time, its (possibly derived) duration or end time and its position on the screen (for non-audio events). The way the information specifying an event is expressed in each of the paradigms is illustrated in Fig. 4.14.

- In the structure-based paradigm, the timing and position information are explicitly recorded together in the structure with (a reference to) a media item.
- In the timeline paradigm a media item is selected and assigned a position on the screen by the author (often by direct manipulation) and its start and end times specified via the timeline.
- In the flowchart and script paradigms a command refers to an object to be placed on the screen and at some later point in the script another command removes the object from the screen (where an audio item is generally only started).

Links are specified in different ways in each of the paradigms. This is summarised in Table 4.5.

- In the structure-based paradigm, the source component and anchor of the link, along with its associated context, can be specified. Similarly the destination component, anchor and context can be specified. In either case there may be multiple sets of these (although this adds yet another degree of complexity to the authoring process). The transition information can be recorded with the link structure.
- In the timeline paradigm the source and destination contexts of the link are restricted to being everything playing at some point on the timeline. The source of the link is either an anchor or a component on the screen which the end-user can select to follow the link. The source anchor can be defined as lasting for some extent along the timeline. The destination is specified via script command along with any transition information, such as duration or visual effects.

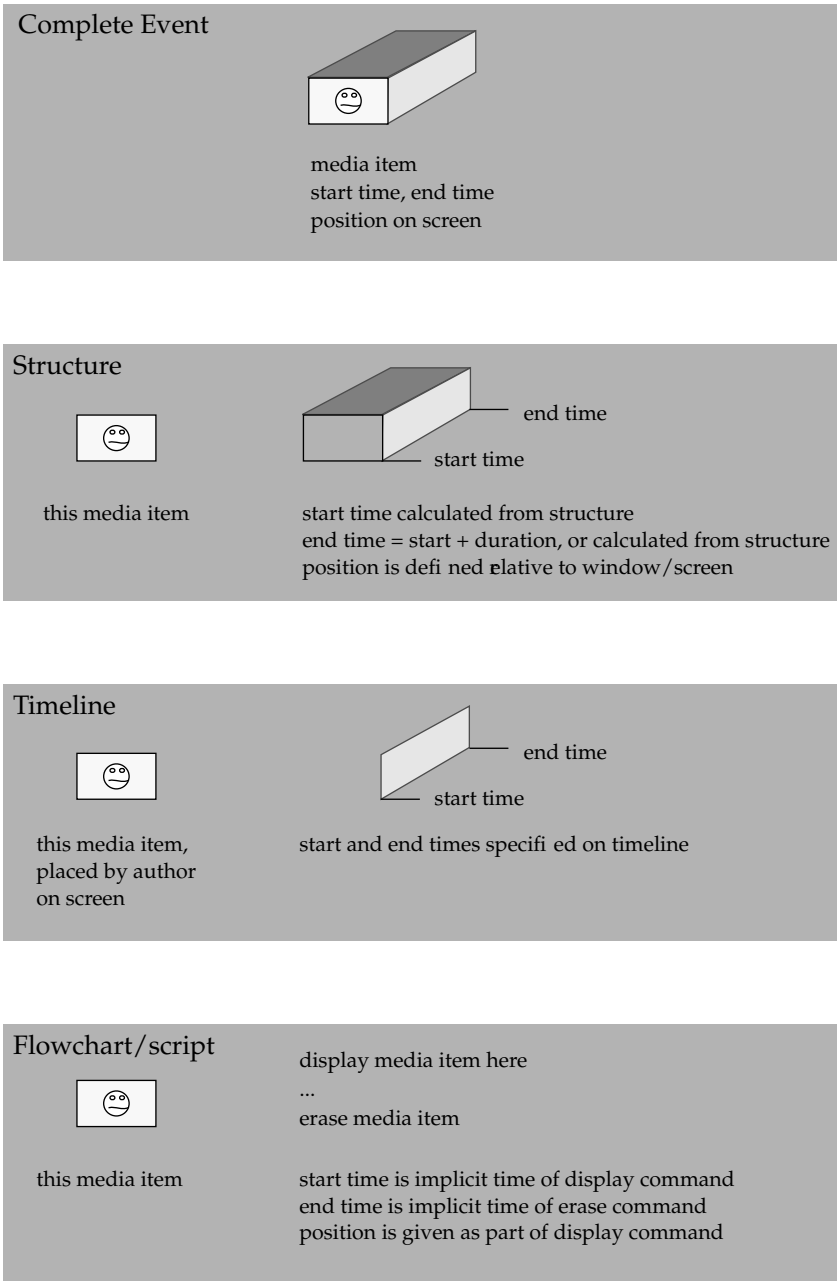


Figure 4.14. Specifying an event in each of the paradigms

- In the flowchart and script paradigms the source anchor is a hotspot object which has an associated script. The script specifies the source context implicitly by erasing some or all of the playing objects. The transition is also part of the script. The destination context is again implicitly defined as the objects that are displayed. There is no special object that can play the role of a destination anchor.

	Source anchor	**Source context**	**Transition**	**Destination anchor**	**Destination context**
Structure	yes	yes	yes	yes	yes
Timeline	hotspot	all events at point on timeline	yes	no	all events at point on timeline
Flowchart/ Script	hotspot	explicitly erase playing items	yes	no	explicitly display new items

TABLE 4.5. Specifying a link in each of the paradigms

Paradigms
Each paradigm emphasizes a different aspect of the creation or visualization of a multimedia presentation. In general, the richer the language the more precise the specification, but the more difficult it is to use. Also, each paradigm has its own emphasis, for example in the structure-based paradigm it is difficult to create scene changes where events at the end of one scene overlap with events at the beginning of the next. This can be perceived as an advantage or as a disadvantage. The trade-offs between the different paradigms are how flexible the behaviour is that can be specified, how easy it is to specify the behaviour, and how easy it is to view the specified behaviour (other than by playing the presentation). This is summarised in Table 4.6.

Property / **Paradigm**	**show narrative structure**	**edit narrative structure**	**navigate narrative**	**show timing**	**edit timing**	**show synchronization**	**edit synchronization w.r.t. objects**	**edit synchronization w.r.t. structure**	**show layout**	**edit layout**	**show interaction**	**edit interaction**	**interaction diversity**
Structure-based	++	++	++	0	+	0	+	++	--	0	-	0	-
Timeline	-	--	-	++	+	++	++	--	+	0	--	--	-
Flowchart	+	0	+	-	--	--	--	--	--	-	-	+	+
Script	-	-	-	--	--	--	--	--	--	-	-	0	++

Key: ++ very good, + good, 0 neutral, - bad, -- very bad/not possible

TABLE 4.6. Properties of authoring paradigms

- Structure based systems are good for viewing, editing and navigating the narrative structure of a presentation, allowing different levels of detail to be shown as appropriate. While not ideal for viewing the timing of the presentation, the structure can be used for editing the timing by allowing (some) timing relations to be derived from the structure. The structure itself gives an ordering of the display of media items. Layout information is specified per event, and an overview of the layout at a particular time is possible only by playing the presentation. Interaction, other than playing the presentation, is restricted to specifying and following links. Links, however, can in principle be defined using source and destination anchors and contexts.
- Timeline based systems have no direct means of editing the narrative structure of the presentation directly, although it can be perceived and navigated as discontinuities of groups of objects along the timeline. The timeline is, however, the best way of showing when objects are displayed on the screen and synchronization relationships among events. It is not necessarily the best way of editing the timing, since although timing of individual objects can be changed, every object has to be manipulated individually, unless some form of structuring is present. Layout is specified per object per time unit, so an overview of all objects at a certain time is possible. The layout, and other properties of an event, can also be shown as a function of time, e.g. in MET^{++}. Interaction specification is even more restricted than in structure based systems, since links are often only jumps to some other point on the timeline.
- The flowchart and script paradigms are comparable in power of expression, where editing and viewing the result tend to be more cumbersome with scripts. Reflecting the narrative structure in the structure of the flowchart, or script procedure calls, is possible but not compulsory. Similarly, navigation of the narrative is only as easy as the procedural correspondence maintained during editing. Timing information, on the other hand, cannot be shown (although as mentioned previously, this may be derived for viewing in a timeline). Layout information is specified per object, generally as part of the command to display the object. An overview of the layout at a particular time is possible only by playing the presentation. The flexibility of the interaction that can be specified is high, and the flowchart tools can help with its specification. Viewing the interaction remains a problem, where the only methods are to run the presentation to check the various possible paths or to navigate the flowchart/script specification.

While each paradigm has its own strengths and weaknesses, we do not wish to choose one paradigm above another, but seek to combine them in a way that takes advantage of their complimentary strengths. The paradigms illustrate different ways of providing similar functionality in a hypermedia authoring environment. They do not, however, provide a solution to the problem of which functionality should be provided. We tackle this question in the following chapter.

5 Authoring Requirements for Hypermedia

In this chapter we discuss authoring requirements for hypermedia. We go through the document model defined in chapter 3 and discuss ways of editing and visualizing its parts. For each document component we present a list of required editing functionalities and give suggestions on how these may be visualized to aid the author. We do not go into design details of a complete user interface, nor do we attempt to combine all the different functionalities into a complete and unified system.

5.1 Introduction

This chapter states the requirements for the next generation of hypermedia authoring systems. We approach this task systematically and state our requirements with argumentation. There is an analogy with word processing, where, while initial systems were diverse, most current systems show great similarities—apparently the available features have reached equilibrium with the requirements. We wish to achieve the same for hypermedia authoring.

A hypermedia authoring system, as an instance of an interactive system, has the components model, view and controller [Burb92]. The model is a document model such as that proposed in chapter 3, the controls are the transactions we wish to carry out on the model, and the views are the ways of visualizing the model to the user. The correspondence between the model and the controls has to be a mutual cover; that is, all the elements in the document model have to be editable, and every editing action should be captured in one or more parts of the document model. Table 5.1 of this chapter, in the appendix, lists the document model elements from chapter 3 along with the sections in this chapter in which they are discussed. We concentrate in this chapter on the control which is needed over the parts of the document model and on the views that can be provided for simplifying the understanding of the state of the model. Where appropriate we discuss specific visualizations for some of the required controls.

The editing environment is divided into 4 layers, Fig. 5.1: the data layer, component layer, document layer and resource layer. These are similar to the Dexter layers [HaSc94], where the data layer corresponds to the within-component layer, and the component layer to the storage layer. The document layer requires the

same input as the runtime layer, but it is a static view of the components it refers to. Calculations, such as the overall timing of the presentation, can be carried out but without actually having to play the presentation. The resource layer is perpendicular to these other layers, in that it contains information external to the component and document layers but can be referred to from them. It is a generalization of the presentation specifi cations in Dexter The layers communicate with each other as indicated by the arrows in the fi gure. We discuss each of these layers in its own section. Note that the runtime engine is not an editor as such, but can communicate with the document editor and the component layer.

While the text is rather extensive, the essence of the chapter can be found in the discussions at the end of each section and, because of the importance of timing and spatial layout in multimedia, in the timing and spatial layout sections of the document layer section. Conclusions are summarised at the end of the chapter.

5.2 Data Layer

The data layer, shown as the lowest layer in Fig. 5.1, contains the data resources external to the document itself. The store of media items, while logically in one place in the fi gure, may be a distributed store. This is the source of the data that forms the basis of the fi nal presentation.

5.2.1 Media items

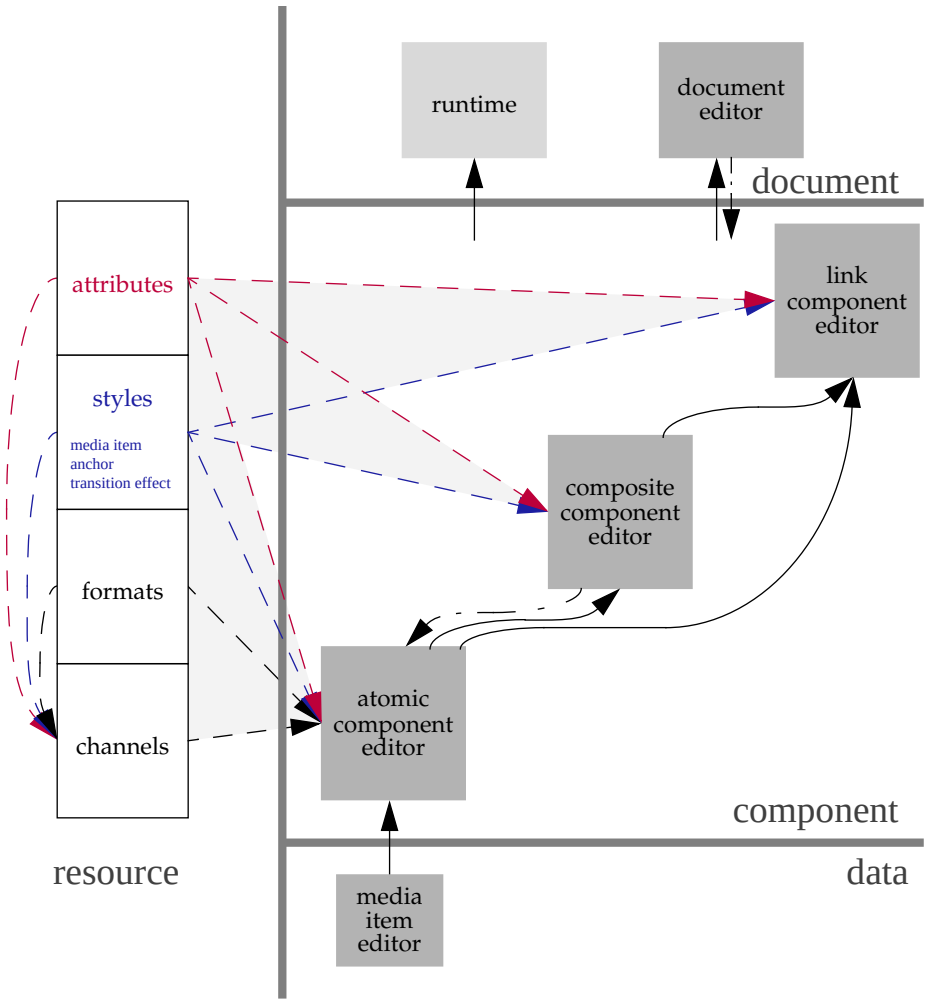
A media item is an amount of data that can be retrieved as a single object from a store of data objects, or is generated as the output from an external process. Media items are the building blocks of a hypermedia presentation—without these there are no objects to display. They can be any of the following media types: text, image, audio, moving image, combined audio and moving image. They may be derived from, e.g., a simulation, or may be three-dimensional, for example virtual reality.

Media items are of interest in the authoring process only in so far as the author needs to select one or more for inclusion in the presentation. It is an essential choice to treat media items of different origin as equivalent within the context of hypermedia authoring. We discuss here the selection of a media item and the specifi cation of some part of it, in other words a reference to part of the media item.

Create media item

No restriction need be placed on the data types that can be processed by an authoring system. The only requirement is that the system can interpret the data format directly or convert it to a format that can be interpreted. The admissible data formats are best stored in a separate resource.

The system may include its own set of internal editors to create, edit and manipulate different data formats, but, given the widely varying editing styles



The arrow from the data layer to the component layer indicates that only the atomic component can refer directly to data. The arrows between the component layer and the document layer indicate that information from every component can be passed to the runtime system and the document editor, and that the document editor is able to edit data structures within any one of the components. Note that only the atomic component uses the format and channel resources.

Figure 5.1. Overview of data flow among the four layers

of both the media type and the author, this is best left to a specialist media-specifi c tool. Whether the media items are created within the authoring environment or externally, we proceed under the assumption that the media items exist and can be accessed and interpreted by the authoring environment.

By defi ning a media item a number of characteristics are defi ned by the data block itself and the corresponding data format, namely the item's intrinsic duration and intrinsic size. These characteristics are medium-dependent, for example an audio item has only intrinsic duration and an image only intrinsic size. A text data format, such as ASCII, has no intrinsic size without any further font size information. If the text data format includes font information but no layout information, then it has a number of potential sizes but no predefi ned aspect ratio.

Select media item
In the context of authoring, media items need to be displayed in some way to allow the author to select them for inclusion in a presentation. This is done in most systems by supplying, at the level of the operating system, a list of fi le names in a dialog box. The problem is that the user has to deduce the contents of the fi le from only its name. A direct coupling of the fi le name with a viewer for that data format would allow the user to select a candidate fi le and then play it to see if it was indeed the one required. An example of an extension to this method is to display miniatures of image and video items. This aids the selection of a particular video fragment or image from the many available.

Edit media item reference
The authoring environment should facilitate the author's task of selecting the appropriate part of a media item either as the block of data to be displayed to the user, or as an anchor value. A media-dependent description of the part of the item is required. For example, in chapter 2 we discuss media-dependent ways of specifying a part of a media item. An editor capable of displaying the media item in its entirety and selecting a part of it is required. An example of a text selection editor is to allow the author to view the complete text and drag out the desired text extent, where the system records the string offset and length. Similarly, for selecting a part of an image the author should be able to view the complete picture and specify a part of it, e.g., by indicating the top-left and bottom-right coordinates with a mouse.

When the data of the item is writable, problems can occur, since it becomes less clear which part was originally specifi ed. Specifi cation of the content needs to be more than a data-only specifi cation. As an illustration, if a text item is editable then contextual information should be provided, such as keywords within the desired string and neighbouring ones just outside the string.

Once the referencing has been carried out the partial media item has its own intrinsic duration and size. It is useful to be able to play the referenced part of the item to check that it is as the author specifi ed.

Discussion
Since the data format of a media item does not play a role in the document model, it is essential to treat the media items included in a presentation as equivalent. We achieve this through the use of an atomic component. Once a media item has been included in an atomic component, the handling of it in an authoring environment is standardized. That is why we advocate the separation of the media data and its envelope—the atomic component. In addition, the atomic component allows the inclusion of the same media item as multiple instances within the presentation.

Special attention should be paid to tools supporting the selection of media items for inclusion in a presentation, in particular, by providing an overview of a large number of items.

When a media item is not custom made for inclusion in a presentation, authors will require tools for modifying the media item to suit their purpose. There are two possibilities: either a new version of the media item can be created and modified, or a part of the existing media item can be selected. Our preference is for the latter as this avoids the proliferation of nearly equal versions and the difficulties associated with intellectual property rights. Selection of part of a media item is not supported in current generation systems. As more pre-created media items become more accessible, in particular via the World Wide Web, then authors will be able to specify the URL to include the media item in their own presentation and only require tools to select part of the item.

In the context of an authoring system, to enhance the reuse of resources we perceive it as useful to store the admissible data formats in a separate resource.

5.3 Component Layer

The component layer contains the objects that an author integrates within a multimedia authoring environment. The components contain data, information about the presentation of the data and information about the data. The *raison d'être* of the components is that they are equal status objects, regardless of the media content. The components are the only means available for recording the specifications of the presentation. In other words, any aspects of the presentation which the author wishes to control must be recorded somewhere within the component layer. The components stored in this layer are atomic, composite and link. The component layer is the central layer in the model and communicates with all other three layers in Fig. 5.1.

5.3.1 Atomic components

Since the data format of a media item should not play a role in the document model, media items included in a presentation should be handled uniformly. We achieve this through the use of an atomic component.

In the previous section we discussed that references can specify a part of a media item. It is the task of the authoring system to use these references in the atomic component in the appropriate places, namely specifying the content and the anchor values, Fig. 5.2.

An authoring system requires, as a minimum, to be able to create and delete atomic components. An atomic component consists of a content, anchors, presentation specifi cation and attributes. We discuss the presentation specifi cation in three separate parts: timing, spatial layout and style.

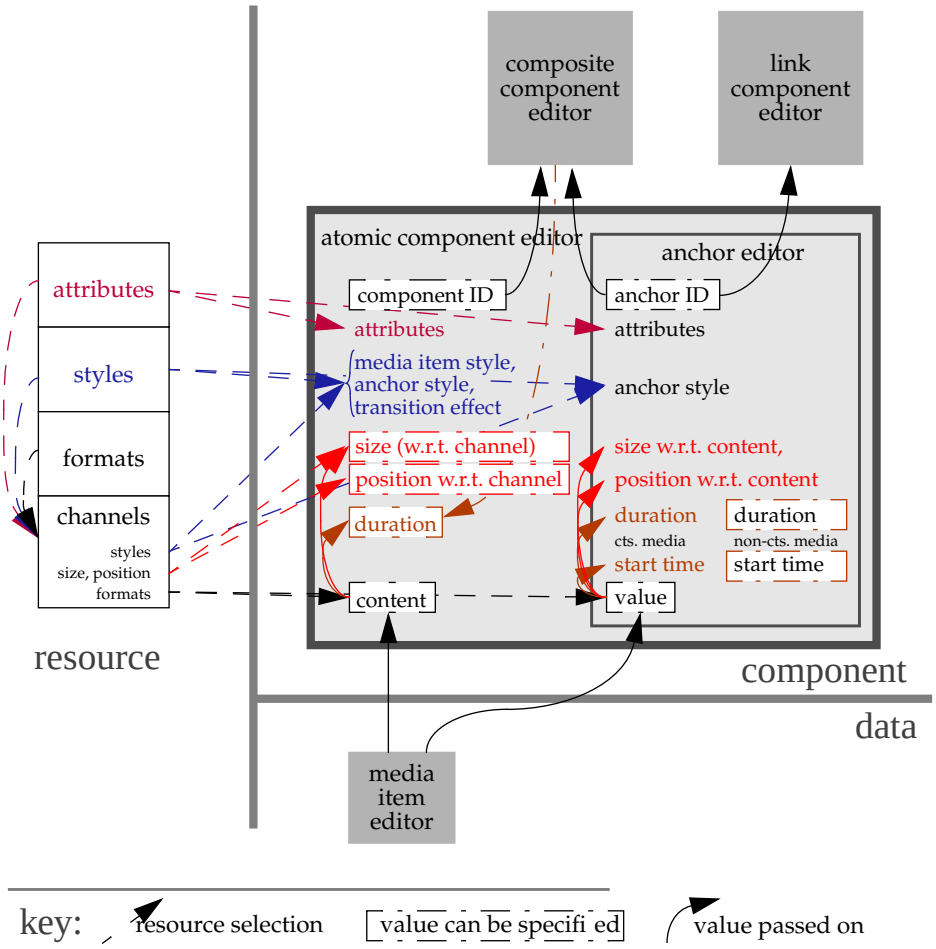


Figure 5.2. Data flow in and out of atomic component editor

Content
The content of an atomic component holds a reference to a complete or partial media item. The data which is referred to from an atomic component is always one block of data of the same media type. The data block brings with it intrinsic duration and size information which can be derived using knowledge of the data format.

The author needs to be able to specify which media item is to be used, and which part of it should appear in the final presentation.

Even though there may be only one data block, this should be selectable from a number of different data formats. At one point one may prefer a presentation in 24-bit colour rather than a 4-bit greyscale presentation. Such a switch between otherwise equal options should be decided at runtime depending on the capabilities of transport and display and of data cost. Therefore we leave open the choice of this part of the data format provided that the parameters of duration and size remain unchanged. This is to prevent the other parts of the atomic component, such as anchors and attributes, becoming invalid, and leaves the place of the data block in the presentation and its dependencies unaltered.

Anchors
Anchors are objects specifying part of a media item and can be used as the basis for creating links and synchronization arcs among components. An anchor consists of an anchor identifier, an anchor value, attributes and a presentation specification.

An anchor identifier must be specified by the author, for example labelling part of a picture with an author-selected keyword, or generated automatically by the system, for example using a generated number or picking a keyword as a name from a text item.

Anchor values should be allowed to overlap, either partially or fully. This is not always the case in current implementations, e.g. HTML. There may be an authoring requirement for changing the start-time, duration, size or position of an anchor value, with respect to the content. While this may be desirable from the author's point of view, it is impossible to supply generic tools for this since the anchor specification is media-dependent. Anchors inherit their duration, start-time, size and position from their specification with respect to the content. These properties can only be modified through scaling the complete content of the component, and thus cannot be specified per individual anchor. For non-continuous media, however, the start-time and duration of the anchor can be specified explicitly.

Anchor styles should be specifiable per individual anchor, which should be selectable from the same resource used for the anchor styles for the atomic component as a whole. An anchor style specified for an individual anchor overrides that specified in the atomic component or channel.

Timing
At this point, we are concerned only with the timing as specifi ed within an atomic component. The duration of the content of the atomic component is the intrinsic duration of the media item reference. The duration of the atomic component should, however, remain editable by the author. For example, by specifying an absolute duration or one relative to the intrinsic duration of the content. Hence, the duration of the atomic component is not bound to equal the intrinsic duration of the media item reference and is stored separately. The playback system needs to resolve how the content is played to conform to the specifi ed duration.

The timing information for an anchor is derived from the anchor value for continuous media items. For non-continuous media the start-time and duration should be specifi able by the author

Spatial layout
At this point, we are concerned only with the spatial layout as specifi ed within an atomic component. The size of the content of the atomic component is the intrinsic size of the media item reference. The size of the component should remain editable by the author.

We consider spatial layout authoring requirements for two distinct cases. The fi rst is that where channels are used for recording spatial information. The second is that of a model that uses no channels, and thus parallels the authoring requirements of timing somewhat closer.

For the channel case, each atomic component is assigned a channel. The author should be able to specify the position and size of the content with respect to the channel. The author should also be able to specify an absolute size or modify the intrinsic size of the content by a scale factor. The playback system needs to resolve how the content is played to conform to the specifi ed size.

For the non-channel case, the author should be able to specify an absolute size or modify the intrinsic size of the content by a scale factor.

Styles
The styles for an atomic component should be selectable from a style resource. This allows the same styles to be applied throughout a presentation and allows changes to apply to all the components using the style. At this point, we are concerned only with the styles as specifi ed within an atomic component. These are: media item style, anchor style and transition special effect.

The media item style is medium-dependent and includes, for example, font for text and background colour for all visual media types.

The anchor style associated with an atomic component applies to all the component's anchors and can be used for, for example, showing the position and size of the anchor value. This is a style resource although separate from that for media item styles.

The transition special effect specifies how the display of a media item is initialised or terminated, for example, "fade-in" or "checker-board". The style should be applicable to all media, but may have medium-dependent interpretations. For example "fade-in" can apply to both visual and audio media, whereas "checker-board" might use a "fade-in" for audio. This is a style resource although separate from that for media item and anchor styles.

A style specified for an individual atomic component overrides that specified in the associated channel.

Attributes

The attributes for an atomic component should be selectable from a resource of semantic attributes. Attributes should also be specifiable per individual anchor, which again should be selectable from a separately held resource. The attributes for anchors and components should be selectable from the same resource since they both describe aspects of the application domain rather than media specific or component specific information.

Discussion

The main advantages of introducing an atomic component are that it hides the data-dependencies of the media item from the rest of the environment, while also allowing the creation of complex components which are indiscriminable from atomic components. This facilitates the creation of complex presentations. We advocate authoring environments that enhance reuse and management of components.

We advocate the support for different levels of quality during presentation, as the selection should be made at runtime depending on the available display and transport resources. Specification of different measures of data quality is largely unsupported in current systems. A reason is that presentations are currently created for a specific end-user platform. We advocate authoring environments which enable the creation of platform independent presentations.

An anchor value is a data-dependent specification of part of the component's content. While it is largely an implementation issue as to how this information is stored, it does have implications for the authoring interface. If the anchor value is embedded, that is defined within the content itself, and the medium is read-only, then new anchors cannot be created. A classic example of this for the text case is HTML, where the anchor extent is defined within the text itself. If the anchor value is external, i.e. specified separately from the data but referring to it, then it needs to be transported along with the data. There would be more flexibility if both methods were possible for all media types: embedded anchors would be available for others to use, and external anchors could be created by non-owners of the material for their own use. Both approaches are supported in Microcosm [HaDH96].

The style and attributes of an atomic component should be selectable from separate resources to enhance reuse of style and attribute resources. An author

should be able to assign styles on a more global basis, and can use the composite component and channel for this purpose. Attributes can be assigned at the composite level, but should be seen as complementary rather than as conflicting.

5.3.2 Composite components

Composite components facilitate the moulding of the narrative structure of a presentation while keeping the authoring process manageable even when the presentation becomes large. This is achieved by supporting the creation of larger presentations, or collections of presentations, and then allowing properties to be associated with these rather than with only the individual components.

An authoring system requires, as a minimum, to be able to create and delete composite components. A composite component consists of a list of children, anchors, a presentation specification, and attributes, Fig.5.3 and Fig. 5.5. We discuss the presentation specification in three separate parts: activation state, timing, spatial layout and style.

Children

An author should be able to build up a larger presentation from sub-presentations and to create groups of presentations. This is captured in a composite component by grouping together a selection of other components. These can be atomic and/or composite components[1] and the grouping can be temporal, for creating a larger presentation, or atemporal, for collecting together a number of presentations. Temporal composition requires the specification of temporal relations among the children. Atemporal composition requires the specification of the initial activation state for each of its children. The composition structure should be able to reflect the narrative structure of the presentation.

From the perspective of the model, a composite is merely a list of its children, but from an authoring perspective it matters a great deal how the author is able to carry out the composition. The two most obvious requirements are to allow already created components to be collected together into a composite, bottom up composition, and to allow an empty component to be created in which further components can be included, top down composition.

For an authoring environment where the compositions can be rather complex some facility is required for viewing the structure. A number of options are illustrated in Fig. 5.4. In (a) the structure is shown as a standard tree. This allows easy interpretation of the structure, but gives no immediate visualization of the timing of the individual elements nor of the presentation as a whole. In (b) the structure is shown using white title bars, where the elements under them are contained within the structure. Timing is indicated by the position and length of the elements. The structure is more difficult to interpret than in (a), but the flow

1. The AHM also allows the inclusion of link components in a composite component. We do not discuss the authoring aspects of this.

of the presentation is more immediately interpretable. In (c) the structure is shown using indentation. This is easy to interpret but does not facilitate the visualization of timing.

Activation state

An atemporal composite requires the specifi cation of the initial activation state of each of its children when the composite itself is played. For example, two of three children of the composite are initially active, of which one starts playing and the other is displayed but is in a paused state.

The composite as a whole has no timing information, since each of its children has its own timing information and they are not related in any temporal way. Fig. 5.5 shows the editor for an atemporal composite component.

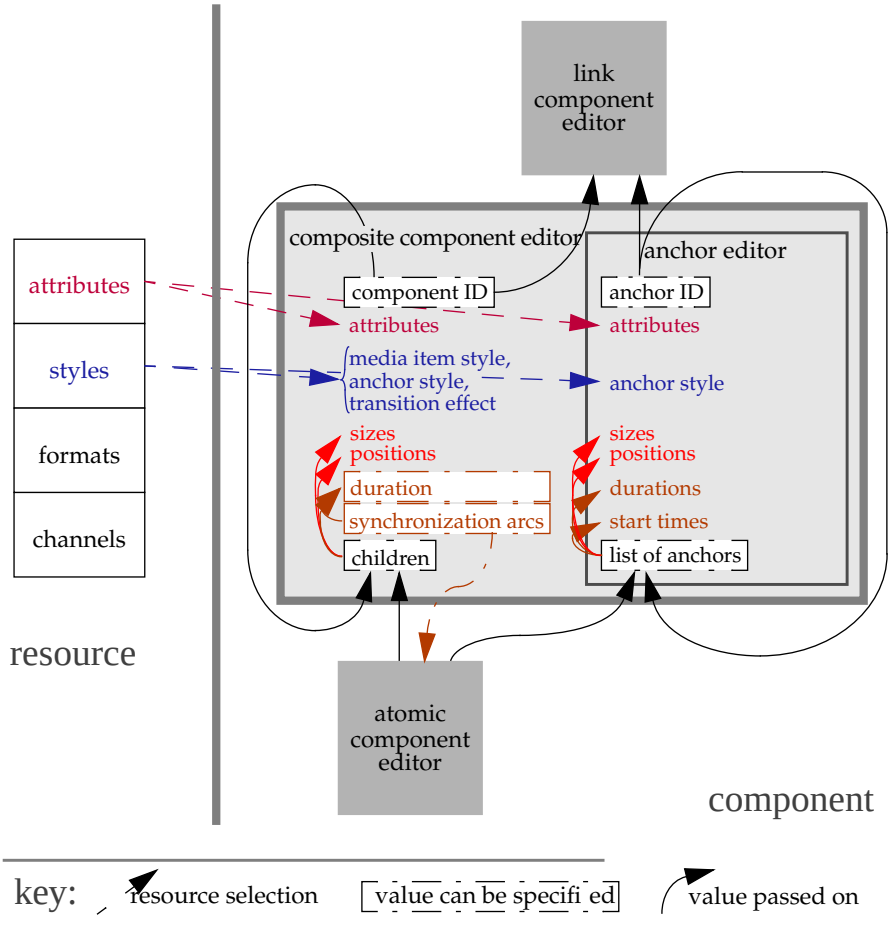
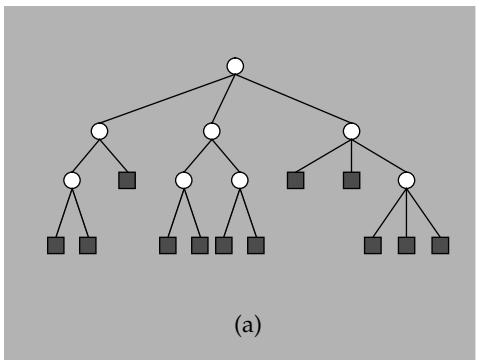
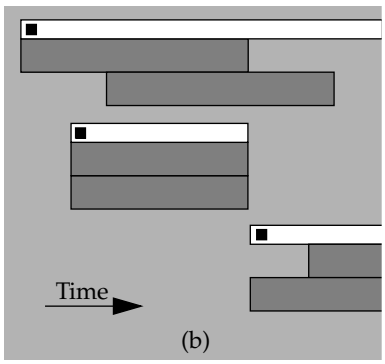


Figure 5.3. Data fl ow in and out of temporal composite component editor

Timing
An author is concerned here with how the timing of a composite component is associated with the durations of the media items and their temporal relations amongst one another. This is captured in a temporal composite including the duration and start times of all its descendants. The duration of a temporal composite component can be calculated on the basis of the durations of its children and any synchronization constraints that exist between them or their descendants. The duration of the composite component should, however, remain editable by the author. For example, by specifying an absolute duration or modifying the total calculated duration of its descendants by a scale factor. Hence, the duration of the composite component is not bound to equal the calculated duration and may be stored separately. It is left to the playback environ-



(a)



(b)

```
Scene Introduction
{
    audio object1 starts at time 0
    {    image object1 starts at time (1 for 5) at position (centre window)
         text object1 starts at time (= image object1) at position (bottom image object1)
    }
    {    image object2 starts after image object1 at position (image object1)
         text object2 starts at time (=image object2) at position (bottom image object1)
    }
    audio object2 starts at time (after audio object1)
}
```

(c)

(a) Depth shows level of hierarchy. Time is not represented.
(b) A white title bar represents structure containing grey boxes and other structures. Time is represented as flowing from left to right.
(c) Indentation shows level of hierarchy. Time is not represented.

Figure 5.4. Visualizing Composition

ment to execute the mapping of the calculated duration to the specifi ed duration at runtime.

Within a temporal composite, synchronization constraints should be defi nable with respect to single events and collections of events (i.e. defi nable between atomic components and composite components). In order to specify a constraint the two components involved in the relation need to be specifi ed along with the appropriate timing relation.

An author should be able to specify the duration of a child component using synchronization arcs in an ancestor composite component. For example, a text item is scheduled to begin and end with an audio commentary, or a sequence of slides should last as long as the accompanying music. The duration of the text item event or slide sequence cannot be found as a specifi c duration in the atomic

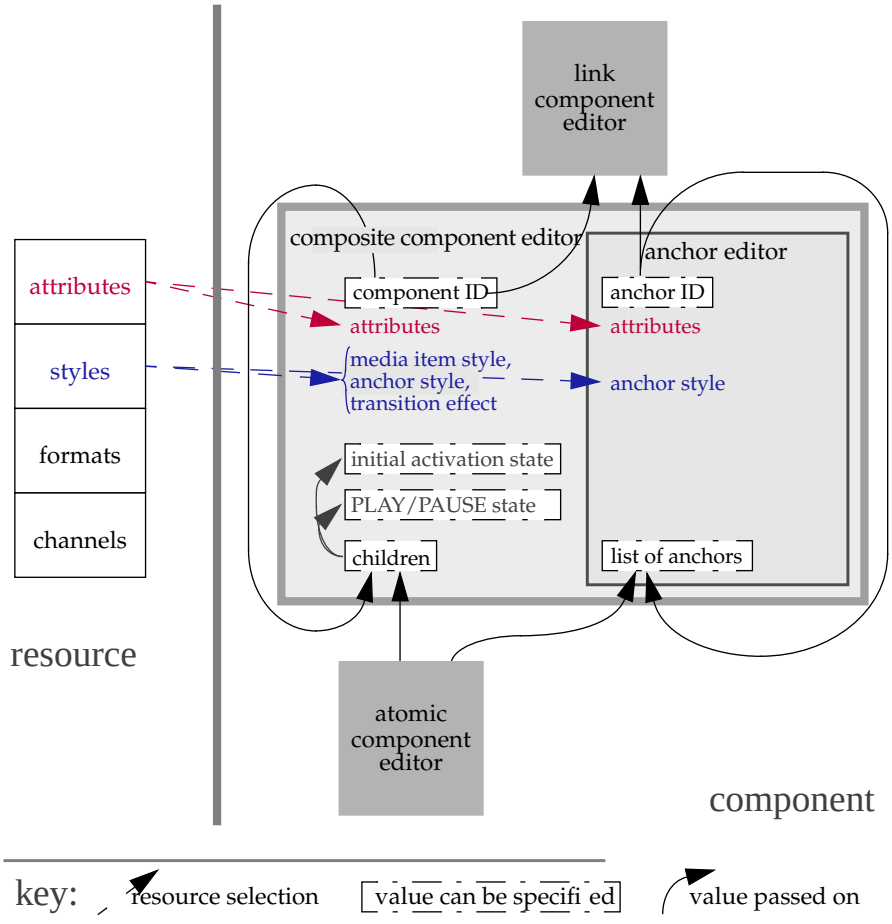


Figure 5.5. Data fl ow in and out of atemporal composite component editor

or composite component duration, but as part of the synchronization arc information in the containing composite.

Spatial layout

We discuss the channel and non-channel cases separately. In the case that channels are used, every atomic component comes with its own spatial layout. Composing these into composite components gives little flexibility in overriding any spatial information, since whatever is specified in the composite cannot override the final channel position and size. The sizes and positions of the atomic components within the channels can be changed, although the authoring requirement for specifying this via a composite component is not immediately clear.

In a model that does not use the channel construct, the size of a composite component can be calculated on the basis of the sizes of its children. Having stated this, the size of the composite may be difficult to calculate, since there is no guarantee that the children are positioned next to each other. The size of the composite component should, however, remain editable by the author, for example by specifying an absolute size or modifying the total size by a scale factor. Hence, the size of the composite component is not bound to equal the calculated size and can be stored separately. It is left to the playback environment to execute the mapping of the calculated size to the specified size at runtime.

Styles

There are three styles which should be specifiable for a composite component: media item style, anchor style and transition effects. The styles should be selectable from the same resource of styles applicable to atomic components, anchors and link components.

Styles should be specifiable at any level of the composition hierarchy. These are recorded in the corresponding composite style specification. The styles recorded in the composition structure may conflict with other styles at different levels of the hierarchy, and it is left to the player software to resolve potential conflicts.

Media-dependent styles apply to all the descendants of a composite, which may not all be of the same media type. The player software has to resolve the application of potentially inappropriate styles, e.g. by ignoring them. Note that this problem does not arise for media item styles since only one media type is involved.

Attributes

The attributes for a composite component should be selectable from a resource of semantic attributes applicable to atomic and composite components as well as to anchors.

Anchors

Anchors are used as the basis for creating links among both atomic and composite components. An anchor consists of an anchor identifier, an anchor value, attributes and a presentation specification.

Specifying an anchor identifier should be the same operation as for an atomic component. Automatically assigning a keyword as the identifier may be difficult since the anchors belonging to the composite may not use identical keywords. If the anchor refers to only one atomic anchor then the identifier could be copied.

The anchor value is given by a list of references to other atomic or composite anchors belonging to descendants of the composite. These resolve to a list of anchor identifiers in descendant atomic components. The author should be given a straightforward way of making this association.

The anchor's semantic attributes and style should be selectable from separately held resources. The attributes should be selectable from the same resource as for components. The style should be selectable from the anchor style resource, where the each anchor may have its own style. An anchor style specified for an individual anchor overrides that specified in the composite component or channel.

Discussion

In authoring a non-trivial hypermedia document the only means of reducing the complexity of the task is to support structural composition. A requirement for this is that the atomic and composite components can be treated in the same way from the author's perspective. We thus advocate systems that allow atomic and composite components to be treated equally.

A further reason for advocating composition is that the composition structure can be used by the author to reflect the narrative structure of the presentation, thus reducing the cognitive load of the author in matching the narrative structure of the presentation to the system-supported representation.

In most current authoring systems, authors are obliged to assign properties to components on an individual basis, in particular timing, styles and attributes. Composition allows the specification of properties on a more global level, relieving the author of repetitive work. We advocate the specification of timing, styles and attributes for composite components.

Composite anchors have not been previously defined, since they only become necessary where links can be created among compositions of multiple data types. In this case, the semantics of the message crosses the media boundaries. The composite anchor construct allows anchors of different data types to be collected together in a structure that the author is able to perceive and manipulate as a whole. We advocate the specification of composite anchors.

Having created a composite component, e.g. synchronizing subtitles with a video, it is useful to be able to include this component in different places in a presentation. This requires not only reuse of data (made possible by referring to

the same media items from different atomic components), but also reuse of the complete composite. The work described in [GaMP94] shows that there is a need for reusing the same composite components in different situations. It is our opinion that authoring support for reuse should be provided.

Composition is perhaps the least commonly exploited aspect in current generation authoring systems. While we have described a number of desired facilities here, as more familiarity is gained with the use of composition in authoring there are likely to be more extensions in the future. An important example is the visualization and navigation of the composition structure, which although a fundamental requirement, is largely unsupported in existing systems.

5.3.3 Link components

Links express relationships among structures and are typically used for navigation purposes. In the previous sections on atomic and composite components we showed that a list of anchors can be made available. The task remaining for the authoring system is to associate a number of these references with each other via a link, Fig. 5.6.

An authoring system requires, as a minimum, to be able to create and delete link components. A link component consists of a presentation specification, attributes, anchors and specifiers. We discuss the presentation specification in three separate parts: timing, spatial layout and style. A specifier consists of a reference to an anchor in an atomic or composite component, a direction and a context.

From the author's perspective, the author wishes to specify when an end-user is able to select a different presentation and how the transition from the running presentation, the source, to the new presentation, the destination, should take place. The author should be able to specify the behaviour and have this recorded as part of the link's properties. In other words, the author wishes to create a continuous presentation where none existed explicitly in the composition structure. The link provides the timing and transition effect information that would otherwise have been recorded in a composition of the source and destination contexts. This virtual composition structure cannot be created beforehand, since it cannot be predicted when the end-user will choose to follow the link.

Specifiers

The specifiers are the information elements where the anchors of the components are referred to from a link. A specifier consists of an anchor reference, a direction, a context and a presentation specification. An author is most likely to want to specify at least one source and one destination specifier, otherwise there is no source or no destination of the link. For each specifier the following are required:

- Anchor reference
  The author needs to be able to select an anchor, via its associated compo-

nent, for inclusion in the link. This may be selected using an interface similar to navigating the composition structure.

- Direction
  For each anchor the author needs to specify whether it is a source, destination or both of the link. An author need not specify the link direction explicitly, but instead a link can be thought of as having source and destination specifi ers. In this case, the author need only create two lists of source and destination specifi ers.

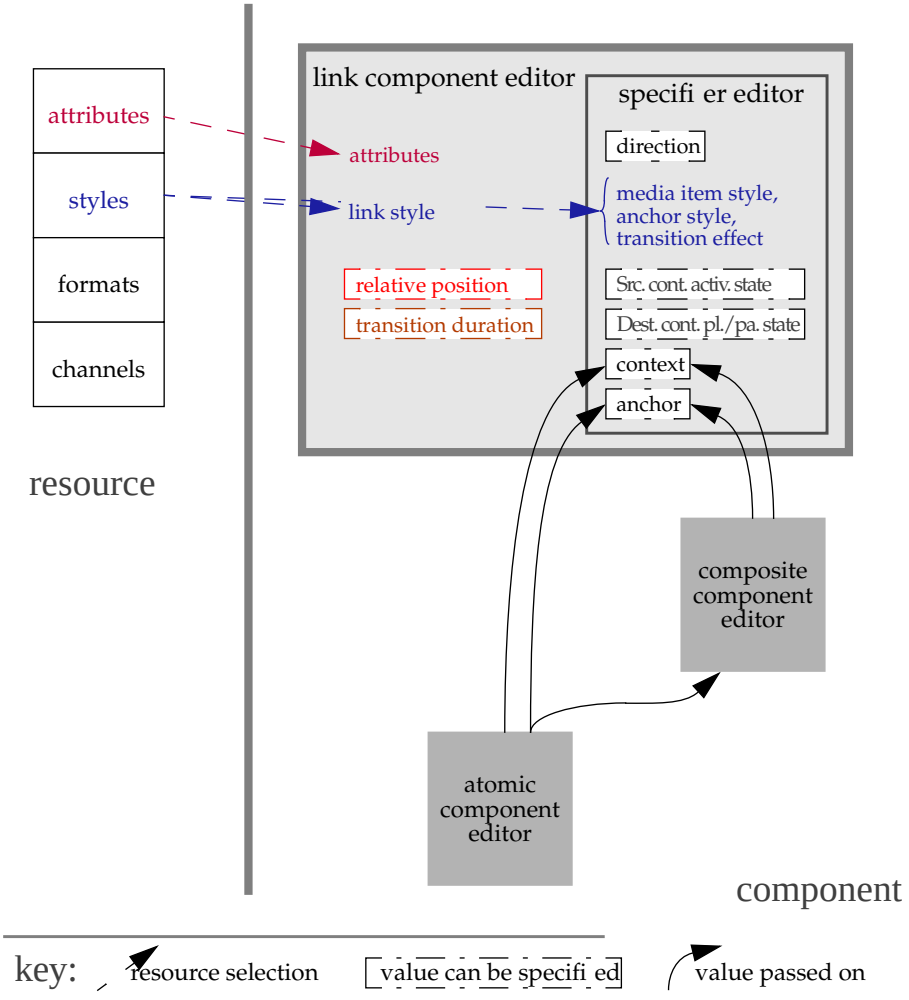


Figure 5.6. Data fl ow in and out of link component editor

- Context
  For each anchor, the author needs to specify the surrounding context of the anchor. This is given by the component containing the anchor or by an ancestor component. Selection can be supported, for example, by an interface similar to that for selecting the component containing the anchor.
- Activation state
  For each specifier which can act as the source of the link it has to be specified whether the context is paused, continued or replaced on following the link. This is the source context activation state. For each specifier which can act as the destination of the link it has to be specified whether the context is played or paused on arrival at the destination of the link. This is the destination context activation state.
- Style
  The author should be able to assign a particular style to an individual anchor, e.g. highlighting it in a particular way when leaving the source of the link. This is stored in the specifier's presentation specification. The style should be selectable from the same external resource of anchor styles as that used for the atomic and composite components. An author should also be able to assign different styles (media item, anchor or transition effect) applying to the destination context of the link, for example to preserve the visual coherence of the presentation by applying the styles used in the source context.

Timing
As part of specifying a link from source to destination context, an author needs to specify how the source context will transform into the destination context. This includes the temporal overlap of the end of the source context and the beginning of the destination context, which can be captured as the duration of the link. The start time of the destination context is valid only at playback time, since it cannot be predicted beforehand when the link will be followed by a user. The author can thus not specify when the transformation will occur.

Spatial layout
Where the destination context has no pre-defined position on the screen, the author should be able to specify its position with respect to the source context. This can be stored in the relative position of the link.

Styles
The style should be selectable from an external resource of styles defining transition effects. These allow the author to describe how the source context transforms into the destination context. This may clash with transition effects assigned to individual components, in which case the author should be able to specify which style should be given priority.

Attributes
The attributes should be selectable from a resource of semantic attributes. Although this should be the same resource as for atomic and composite components, it is likely that the attributes assigned to atomic and composite components describe objects or groups of objects and those assigned to links describe relationships. These allow the author to describe the relationship between the anchors at the source and destination of the link, for example, " is part of", " is a", " is an example of".

Anchor
We do not discuss links to links in the context of multimedia.

Discussion
The main advantage of introducing a link component is that it allows the author to specify multiple routes for the end-user to follow through a multimedia presentation.

While there is a large amount of experience with creating links in hypertext there is almost no explicit treatment of links within multimedia. Editing links within multimedia requires the management of source and destination contexts along with transition information, in addition to the usual source anchor and destination component often seen in hypertext. As yet there are no editors which support the selection of source and destination contexts or transition information explicitly. We believe that such editors need to be implemented. The first attempts to supply such support are likely to be cumbersome, and real use by authors is needed to find useful short-cuts to specifying all the required parts.

Rather than requiring an author to create links on an individual basis, a higher level approach can be achieved by creating links among abstractions instead of among concrete components. An example is given in [HoSA89], although without a description of the facility, and another example is implemented as generic links in Microcosm [FHHD90], [HaDH96]. The design of a system to help automate the authoring process, including the generation of links, is given in [WBHT97]. We advocate authoring support for high-level creation of links.

Although link specifiers support the specification of bidirectional links, in our experience it is rare to use these in multimedia. This is because of the asymmetric relation between source and destination: the user can select only one anchor at the source, whereas the destination can be a complete scene consisting of a number of media items. In the case that the destination of a link is a complete scene with no highlighted anchor, following a link back from the composite component is not possible since no visualization of the anchor is available. Although hypertext links can usefully be bidirectional, we believe this is not necessarily the case for multimedia. While we do not wish to suggest that bidirectional links have no place in multimedia, we do not demand that these be supported explicitly in an authoring environment.

The source and destination contexts of a link require a temporal relation for the case that the link is followed. We state that there is a requirement for specifying the duration of a link.

The styles and attributes applicable to a link component should be selectable from separate resources to enhance reuse of style and attribute resources. We advocate the support of specifying relevant styles for the source and destination of a link and assigning attributes to the link as a whole.

In summary, linking in hypermedia is similar to linking in hypertext, in that it allows the specification of a relation between source and destination. In hypermedia, however, it is more than this, requiring the specification of how the presentation will behave when moving from the source to the destination of the link. The author should be supported in specifying parts of the link, and in being given an overview of how the link will behave on traversal. Where possible, higher-level creation of links should be supported.

5.4 Document Layer

In the previous section we discussed timing and spatial layout in terms of an author's requirements for the atomic, composite and link components. We now discuss authoring requirements from the perspective of the document layer. The document layer provides a view onto the component layer allowing the author to deal with single aspects of the presentation using specialist editors, in particular the presentation's timing, spatial layout and link management. The results from these editors are not recorded in separate data structures but as part of the information stored in the components. While it is possible to create a hypermedia presentation using only the component editors, these do not give the author any insights into important aspects of the presentation, such as temporal and spatial layout. We advocate aiding the author by supporting the assignment of this information at the document level.

5.4.1 Timing

In the section on components we pointed out where different aspects of timing are to be found within the component layer. We confine ourselves here to the timing of the document, showing how the different timing specifications determine the timing of the presentation as a whole. This depends on the durations of events and the timing relationships among events. These in turn are based on the durations of atomic components and how these are combined into composite components. While most timing constraints in a presentation are specified beforehand by the author, following a link causes the destination context of the link to be started-up at a non-predictable time. While this cannot, and should not, be prespecified by the author, the duration of the transition can be specified.

We discuss the requirements of timing specifications, including the duration and start time of an event, synchronization relationships and higher-level edit-

ing actions. In order to facilitate the specifi cation of the timing of the events making up the presentation it is useful, if not essential, to provide an editable visualization. The temporal layout of a presentation is best visualized by means of a timeline. This allows the author to see which events occur when throughout the presentation. We illustrate the authoring requirements with example visualizations where appropriate.

Duration of event
The event associated with an atomic component has a duration. As discussed in section 5.3.1, this can be the intrinsic duration of the content, specifi ed by the author to be a relative or an absolute duration, or, as stated in section 5.3.2, determined from the synchronization arcs in the surrounding composition structure. The author should be able to specify the duration using any of these methods, see what the duration is and which method was used to specify it. Fig. 5.7 shows a typical timeline showing the durations of events and which nodes have intrinsic durations.

Start time of event
The event derived from an atomic component has of itself no explicit start time. This is captured in the composition structure via the synchronization arcs. For an atemporal composite the children are not related in any temporal way, so that the start time of any child can be determined only at run time. A temporal composite or an atomic child of an atemporal composite is a presentation. The synchronization arcs and the durations of the atomic components determine the timing of the associated children and thus the start times of the individual events.

The start times, and thus the synchronization constraints, need to be specifi able and viewable. A timeline is the ideal way of showing constraints between events in the presentation, Fig. 5.8(a). Another way of viewing is required for constraints between structures, for example as in Fig. 5.8(b) and [Acke94].

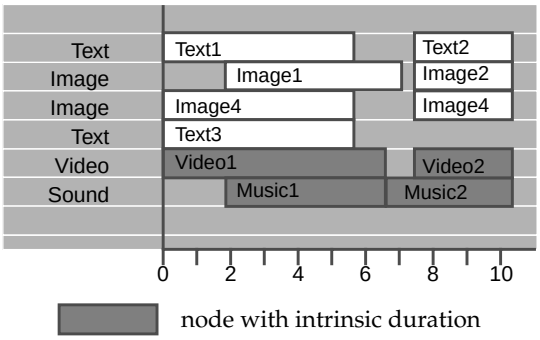


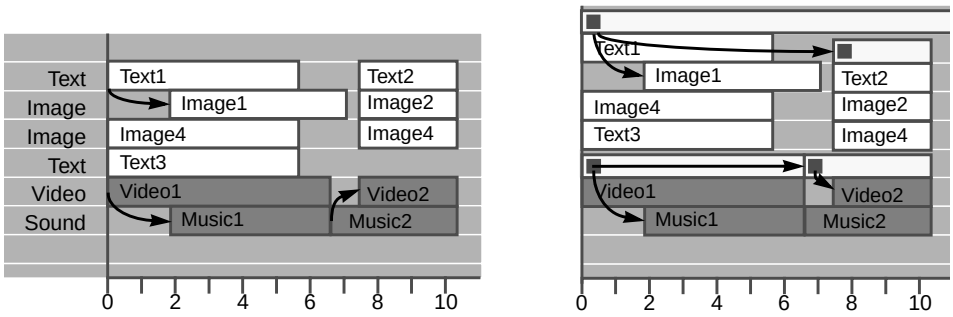
Figure 5.7. Generic timeline

Derive timing constraints from structure
In providing editing facilities for timing we seek to alleviate the author of as much trivial work as possible. Timing does not necessarily have to be defi ned for each individual event, but can be derived from the surrounding structure. For example, if an author groups items to be displayed at the same time they could start and fi nish simultaneously by default, allowing overrides to be specifi ed as required. Examples of deriving timing from structural constraints can be seen in [Acke94] and [HaRe94]. In combination with using durations derived from the media items themselves, the author need only be obliged to specify the duration of a collection of non-continuous media.

Duration of a link transition
An author needs to be able to specify the duration of following a link from the source context of a link to the destination context. A timeline view containing only the source and destination contexts allows the duration of the transition to be visualized and specifi ed without afecting any other timing.

Tempo
As well as specifying the durations of events, the tempo, i.e. the rate at which the presentation is displayed, should be specifi able. This is not an editing action on the presentation itself, but on the rate at which it is played.[2]

Where the tempo of the presentation can be changed, some way of indicating this is needed. There are two possibilities for incorporating this in a timeline representation: i) preserve the scale of the timeline and vary the length of the events, as illustrated in Fig. 5.9(a), or ii) change the scale of the timeline and preserve the length of the events, as illustrated in Fig. 5.9(b). A third possibility is used in



(a) Constraints between events

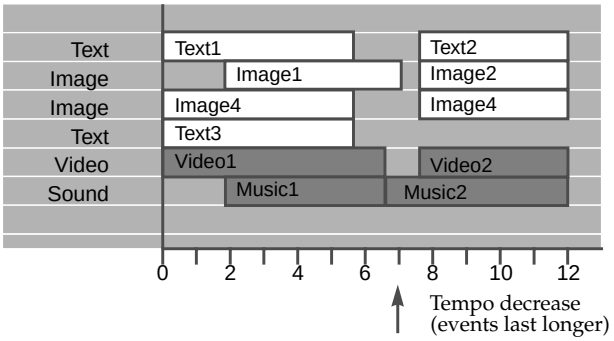
(b) Constraints from and between structures

Figure 5.8. Synchronization Arcs

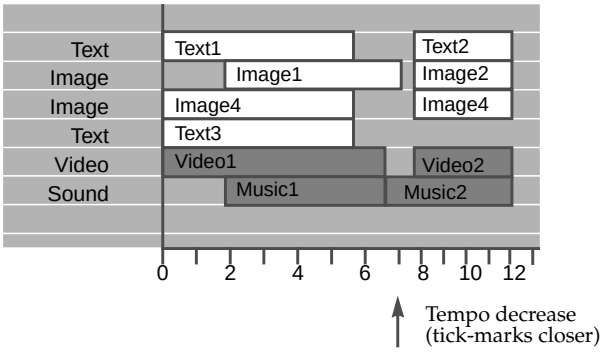
2. Note that we are unable to store this as part of the AHM.

music scores where changes to the tempo are specifi ed but not visualized, so that neither the length of the timeline nor the length of the event is changed. In a visualization for multimedia it should be left to the author to choose the representation.

Applying temporal transformations throughout hierarchy
While minimum authoring requirements can be met by supporting the specifi cation of all parts of a presentation, we strive to reduce the authoring effort. One means of achieving this is to use the composition structure of a presentation for delimiting boundaries over which a particular operation is carried out. For example, in [Acke94] a temporal transformation can be applied to a composite structure and the effect is propagated throughout its descendants. This allows several durations to be edited using only one action.



(a) Extent of events changes



(b) Scale of timeline changes

Figure 5.9. Changing tempo

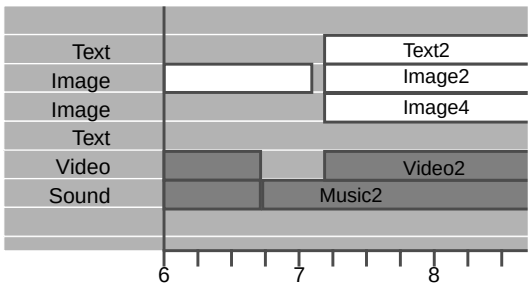
Synchronization specification
Synchronization arcs allow the individual specification of timing constraints, which can be cumbersome when a large number are required. Where many similar operations need to be carried out, a considerable amount of authoring work can be saved by providing higher-level operations. Tools should be provided to support the author in specifying the constraints at a higher-level—the results of which are still stored as individual synchronization arcs.

An example is to highlight individual words in a written text as each is spoken in a commentary [HaKe97]. Anchors can be specified, in both text and audio versions, corresponding to each word. Synchronization arcs are needed to specify that the duration of the highlighting of the word is to last as long as the audio fragment. Two synchronization arcs are required per word. Authoring effort can be spared if the text and audio components are selected and a command such as " match length of anchors"is carried out. An example at a different level in the document structure is where a selection of audio fragments are to be played for the duration of a number of video clips. Each audio fragment should be faded out if too long, or repeated if too short. The two containing composite components could be selected and a command such as " match length of nodes"could be carried out.

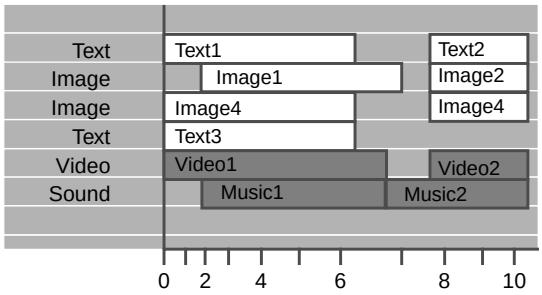
Timeline Navigation
The timeline for a single presentation may become long and unmanageable so that there needs to be some way of changing the scale of the timeline. Possible visualizations are shown in Fig. 5.10, for example in (a), a simple zoomed in view of part of the timeline, or in (b), a fish-eye view

Discussion
At the document level, timing is a crucial aspect in authoring a hypermedia presentation. The complications of timing include the intrinsic dynamic aspects of the media items, the bandwidth needed for delivering them at sufficiently high quality, and the difficulties of executing temporal specifications. The timing of a presentation is so important that this is often used as the authoring paradigm, see for example the timeline based systems described in chapter 4. While this 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.

Timing of individual events can be specified by allocating individual start and end times, or by using timing constraints between events. We advocate the latter, since it reduces the authoring burden when making changes to the presentation. An example of specification of timing via constraints is implemented in Firefly [BuZe93]. The system was not designed to be a complete multimedia authoring system so that while they have a representation for constraints, this is not translated into a timeline representation. We propose that both should be available to the author.



(a) Timeline zoomed in (4x)



(b) Fisheye view

Figure 5.10. Navigating timeline

Constraints should also be specifiable between groups of events. This requires the composition structure and time to be displayed in the same view. An example of this is shown in [Acke94], although temporal constraints were not implemented in the system. We propose that such a facility be available to authors.

We advocate the derivation of event start time and duration from structure as much as possible. This saves the author work by not having to specify timing details for every event.

We advocate the provision of authoring tools for specifying timing constraints at a level above that of single synchronization arcs. This saves the author work by creating multiple synchronization relations with a single command.

Explicit transitions between source and destination contexts of a link are not yet supported in authoring systems. While we discussed a single duration of a link transition, in the case that there are multiple source and destination contexts for the link there is an authoring requirement for specifying the duration of the special effect for each context separately. There is currently little experience in using explicit link transitions, and so it may be that this is more complex than needed for most cases.

In summary, the timing of events in a multimedia presentation is the key characteristic of a multimedia presentation. It should not, however, be used as the basis for authoring but, as far as possible, be derived from a structural representation.

5.4.2 Spatial layout

In Section 5.3 on components we pointed out where different aspects of layout are to be found within the component layer. We confine ourselves here to the spatial layout of the document, showing how the different layout specifications determine the layout of the presentation as a whole. This depends on the layout of events and the spatial relationships among events. These in turn are based on the layouts of atomic components and how these are combined into composite components. We now bring these together and show how they determine the layout of the presentation as a whole.

There are two possible approaches to specifying position for atomic components, either through assigning a channel to each atomic component, or by specifying position information via the composite components. For the purpose of specifying authoring requirements we discuss both approaches. When a link is followed the spatial overlap between the source and destination contexts can be specified.

We discuss the requirements of layout specifications, including the size and position of an event, how these vary as a function of time, and higher level editing actions. In order to facilitate the specification of the layout of the events making up the presentation it is useful, if not essential, to provide an editable visualization. Whether the layout is pre-specified via channels, or edited per event, is irrelevant for the display of the layout of the events. We illustrate the authoring requirements with example visualizations where appropriate.

Size of event

The event associated with an atomic component has a size. As discussed in Section 5.3.1 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. The author should be able to specify size in relation to other objects, for example, increase the font size of a heading until it is the same width as an image[3]. Visualizing the size of a single event is trivial—the media item just has to be displayed on the screen.

Position of event

The event derived from an atomic component has of itself no explicit position. When channels are used, an atomic component has an associated channel which specifies an area in relation to another channel or window. Position information

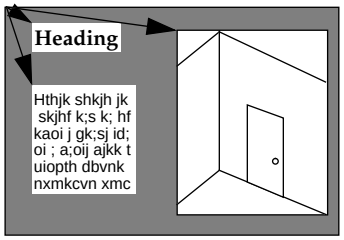
3. Note that while an author may wish to define position or size in terms of constraints, the AHM has no spatial equivalent of a synchronization arc.

for the content is specifi ed in relation to the channel. If channels are not used, then position information can be captured in the presentation specifi cation of a containing composite component. The author needs to be able to state which event an event should be placed relative to, and what the relative position is, for example, a subtitle is constrained to be placed at the bottom of a video and centred with respect to it[3].

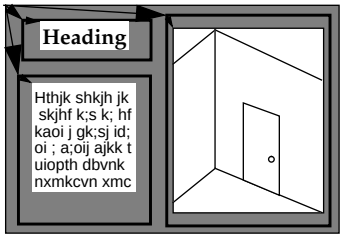
While specifying the position of a single event can be done in a straightforward manner (e.g. by placing the object in the correct position on the screen, typing in coordinates or assigning it to a channel), the author is aided in this placement process by being able to see the positions of the other currently playing objects.

Visualizing the size and position of one event is trivial. Providing a visual overview of the layout of events playing at any one time is a minimum requirement, shown for example in Fig. 5.11(a) and (b). The author also needs to have some overview of the layout of the complete presentation. The layout of a presentation can be visualized by running the presentation, allowing the author to see where objects are displayed with respect to one another throughout the presentation. This method, however, is not a good way of getting an overview of the layout throughout the presentation. Visualizing the position of the objects with respect to time requires three dimensions. The three dimensions can be projected back to two [OgHK90].

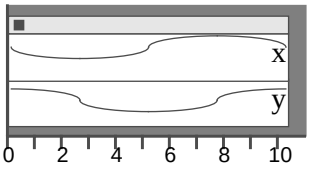
All position information may vary with time. When an object's position is a function of time this is referred to as the path of the object. This can be specifi ed



(a) Position with respect to window



(b) Position with respect to chan



(c) Position changing with time

Figure 5.11. Spatial layout

by a start point, the trajectory to be followed and the, possibly varying, speed of following the trajectory. Alternatively, it could be given by specifying the position of the object at each point in time. The path traced out by an object should be able to be visualized during authoring as well as at runtime. For example, the movement of an object over time can be visualized as horizontal and vertical positions along a timeline, Fig. 5.11(c), [Acke94].

An author should be able to specify the position of the events in a presentation while being able to view the other currently playing events and to be able to see how the layout of the presentation changes with time. An author also has to be able to specify the position of events changing over time.

Position of transition

Although transitions are normally experienced as temporal transformations, the author should also be able to specify a spatial relationship (chapter 7 of [Bufo94]), e.g. two overlapping images could blend into one another in their area of overlap. The overlap of the source and destination contexts could be specified using a spatial constraint between the source context and the destination context. This is probably most useful for the author when specified as part of a link transition dialog. On the other hand, the spatial overlap may not be specified explicitlybut taken as the spatial placement already specified for each source and destination context.

A layout view containing only the source and destination contexts allows the spatial layout of the transition to be specified without affecting any other layout. The visualization would be similar to that in Fig. 5.11(a), where there would be only source and destination contexts, removed from their containing composite components, displayed within the same spatial coordinate system.

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. Channels are a means of doing this through pre-defining areas in a window. Rather than having to specify the position for each individual event, the author need only assign the event to a channel. If the size or position of the channel is changed then the change applies to all items displayed via that channel. The use of channels also makes creating a consistent layout easier because events assigned to a channel are displayed at the same, approximate, position. The channel supplies more than a default position, since it constrains that the event be displayed within the channel boundary. Authors should be able to specify how the size and position of the event relates to the channel as follows.

- The position of the event in relation to the channel, for example, centred at the top.
- Whether the event is scaled to the size of the channel or retains the dimensions specified by the atomic component. The latter is undesirable for retaining the presentation's scalability. The advantage of the former is that,

regardless of the stored size, all the events played via the channel will appear the same size.
- If the event is scaled, whether it retains its aspect ratio. For text this is less important, but for images they will become distorted.

Channels themselves should be able to be grouped into a *layout*—a template containing a number of channels. Layouts should be able to be included in other layouts, thus providing a hierarchical structure to the layout of the presentation. Channels should have their extents and/or positions defined with respect to one another or with respect to a layout.

Derive spatial layout from structure

The author can be spared effort by specifying editing actions at a high level. While deriving temporal information from structure was an example of this, it is more difficult to state how the spatial information can be derived. If the structure is used for foreground and background objects (for example placing different pieces of text on top of a picture) then the lower levels of structure are displayed on top of the higher levels. As far as we are aware, there is no system which supports the derivation of spatial layout, and we can supply no arguments for providing such a facility. This may be an indication that solving the constraints in two spatial dimensions is so much more complex that it is better left to the author.

A combination of using structure together with a channel layout could be made, so that components specified at a high level in the hierarchy, e.g. a heading or a background, are assigned particular channels from the layout.

Applying spatial transformations throughout hierarchy

The structure of a presentation can be used for delimiting boundaries over which a particular operation is valid. In the spatial case, a transformation can be applied to a composite structure and the effect propagated throughout its descendants. For example, a composition of a video with its accompanying subtitles could be scaled to fit a new window. This type of transformation has not been implemented as far as we are aware.

Where channels are used, these transformations should be applied to the layout structure, captured by the hierarchy of channels, rather than the document composition structure.

Spatial layout navigation

In the case of timing, an author needs to be able to visualize the overall timing of the presentation as well as detailed synchronizations between events. In the case of layout, however, the author needs to compare different layouts with each other. For example, when one picture is replaced by another the replacement should appear exactly in the same place on the screen. In both examples it is actually the timeline that the author requires to navigate, and not the layout itself. The requirement for the author is not that they should be able to navigate

one particular layout, but that the presentation can be viewed at (at least) two points on the timeline at once in order to compare layouts.

Discussion

Spatial layout and timing are both extremely important aspects in authoring a hypermedia presentation. While the complications of timing are to do with the intrinsic dynamic aspects of the media items and the difficulties of carrying out the temporal specifications, spatial considerations are complicated by their two-dimensional nature. Not only are the sizes and positions required in two dimensions, but there is no screen-dimension left for providing an overview to the author. Other means then need to be found.

In many current authoring systems, in particular the structure-based, flowchart and script systems discussed in chapter 4, position per event is specified independently of any other events. This makes it difficult to get the positioning correct without playing the result. A minimum authoring requirement is that the author is able to see the positions of co-occurring events while editing the position of an event. A time-line based view, where the position of the event is editable, is a means of providing this functionality.

In all the systems discussed in chapter 4, the position of an item is defined with respect to a containing window or the screen. This has as consequence that spatial constraints between events, or groups of events, cannot be specified. The constraint approach would allow the position, or size, of an event to be changed and have other events follow it. It is unclear as to why systems, so far, support only the specification with respect to a window. It could be that this is sufficient for most purposes. On the other hand, no system has yet provided the opportunity for authors to specify constraints with respect to other events or groups of events.

We advocate the use of channels for specifying consistent layout in a presentation. Channels were introduced into a document model for hypermedia not only for the specification of layout, but also for the grouping together of media items that use similar resources. Resource control of a document retrieved over a networked environment can thus be controlled for the complete presentation rather than for each individual media item. A more in-depth discussion of resource control lies outside the scope of this thesis.

A disadvantage of using channels for spatial layout is that spatial constraints cannot be specified directly between events. For temporal information, however, we do advocate the use of synchronization arcs directly between events. The difference between the spatial and temporal cases is based on the notion that time is typically longer than the screen is large, in other words a larger number of events can be placed along the timeline than can be placed on the screen at one time. That is, the advantage of having consistent layout for different groups of events outweighs the disadvantage that if the position of one event is changed then all the other positions need to be changed. The same action of consistently

dividing the screen space into reusable units would be to divide the timeline up into equal length chunks. This is not generally useful for a presentation consisting of events of different lengths. For the temporal case, allowing constraints to define the relationships means that the effects of editing actions can be applied easily throughout the timeline without any author intervention. For the spatial case, the layout changes have easily overseeable effects with no need for constraint solving.

The position of a component can be a function of time, so that the associated media item traces out a path. Visualizing the path is important though difficult. For example, horizontal and vertical projections along a time axis do not give an intuitive feel for how an object will behave on the screen, nor how all the objects will appear in relation to one another. The only other way of comparing paths of objects is to play the presentation. We advocate the provision of a visualisation for an event's path, but are unable to recommend a particular approach.

As well as allowing the position of an object to change with time, the size of an object should also be able to change with time. Size changes over time can also apply to a transition, where, for example, as the destination context appears it expands to fill its allocated space. Neither of these possibilities is, as far as we are aware, supported in current authoring systems. We advocate the support of changing object size in an authoring environment.

Explicit transitions between source and destination contexts of a link are not yet supported in authoring systems. While we discussed the spatial relation of single source and destination contexts, it may be that there are multiple source and destination contexts, and that for each destination context there needs to be information specifying the position of its display. There is currently little experience in using explicit link transitions, and so it may turn out that this is more complex than needed for most cases.

Specifying the position of events in a multimedia presentation is, along with the timing, one of the most fundamental requirements. We have shown in our discussion that there are a number of important authoring requirements which should not be overlooked or dismissed as trivial.

5.4.3 Link management

A hypermedia author is concerned with the creation of a presentation narrative, which, while supported by composition of components, also requires the specification and maintenance of links among these components. An author is not only required to create individual links among components but also to ensure that the possible paths through the presentation are meaningful to the end-user. During the process of creation the author requires some means of ascertaining which links are not yet complete, which have been verified and which still need to be verified. A complete link maintenance environment needs to be provided for the author to carry out the various tasks required. We do not attempt in this section

to provide a complete list of tools for link management, but instead provide some examples of support that should be considered for inclusion in a system.

Find incomplete links

The author is aided by providing a list of links which are syntactically incomplete, i.e. those that do not possess one or more of: source anchor, source component (implied if a source anchor has already been given), source context, transition duration and effect, destination anchor (optional), destination component (optional), and destination context. The author needs an independent list from which a link can be selected then edited to the author's satisfaction before choosing another incomplete link from the list.

Check complete links

Having created a number of links, the author requires to check them for semantic validity and presentation aspects. For the case of the link the latter are the duration of the transition and a special effect.

The author can be aided by the system providing a list of the links in the document which can be annotated with a note of whether a link is satisfactory or not, and a means of generating a list of links the author still has to check. Sublists of this sort may also be useful, for example, list all the links with the selected component as part of the destination context.

Find unlinked components

The author should be able to ask for a list of the composite components which, given the document structure (both composition and link structures), can never be reached from the initial starting point of the presentation. It should then be left to the author to act on this information.

Discussion

A hypermedia author requires to specify and maintain links among separate presentations. 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 advocate the provision of a link management system to aid the author in organising the creation and maintenance of links. One method of providing this functionality is to support the creation of author-specifi able lists of links which can be annotated with status information.

5.4.4 Presentation Control

Having created the various aspects of the presentation, an author needs to preview what has been created. This can be by playing all events in the presentation that are displayed at the same time, but also by viewing subsets of the events. In order to keep playback overhead to a minimum, previewing should be able to be done on any atomic or composite component, allowing complete scenes to be

viewed, or only those events which are part of a local context. The author should also be able to play all events starting from a particular time, and be able to fast forward and reverse the presentation.

While the presentation is playing, the author should be able to see which parts of the editable representation are currently playing. For example, by highlighting the currently playing events in a timeline or composition hierarchy. Fig. 5.7 provides an illustration of a timeline representation. For the case of viewing a subscene some means is needed of showing that not all the events are being played. This would require a combination of structure and timeline view, e.g. Fig. 5.8(b).

Discussion

While an author requires to play a presentation to check through it, this should not be an independent compile/run cycle that requires large amounts of the author's time. A more efficient use of the author's time is to allow the author at any point in the authoring process to play any part of the presentation. While the presentation is playing, the author should be able to select any media item in it and inquire where this is included in the presentation's structure. In other words, the editing environment should be able to pass any (consistent) part of the presentation's specification to the playback system, and the playback system should be able to communicate to the authoring environment which parts of the presentation are actually playing. The author then has the most control and flexibility of when and how much to play of the presentation under construction.

5.5 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 include data format, style, channel and attribute.

5.5.1 Data format

A data format resource is needed so that a playback system knows how to present the data, and also so that spatial and temporal information can be deduced from the media item for use in other parts of the system, for example display in a timeline view. SGML, and thus HyTime, for example, specify the data format using internationally recognised data descriptions. While these need to be specified, the author should be supported by the environment by recognizing standard formats as much as possible and not demanding that the author specify the format for every media item in the presentation.

Discussion

External information is needed in order to be able to interpret the data format used by a media item. This is best assigned via a resource, otherwise a universally parsable description is needed in every atomic component.

5.5.2 Styles

Style information allows the display characteristics of events to be described. The situation is similar to word processors where a number of styles can be defined and applied to all the text annotated with that style—when the style is changed the appearance of all pieces of text with that style changes. From an author's perspective, the styles are selected and applied rather than edited or created. We propose providing higher-level editing notions for assigning style information, allowing the author to apply desired changes to large groups of objects at once. We describe each of the three style types briefly before discussing their authoring requirements.

- *Media item style*
  Media item styles specify display characteristics for media items. They may apply to multiple media types, for example background colour, or to only one, for example line spacing for text. Style information for media items can be stored with the atomic component referring directly to the media item, with the channel associated with the atomic component, with an ancestor composite component, and with a link specifier (applicable to the context when it is used as a destination).
- *Anchor style*
  Anchor style information specifies display characteristics for anchor values. This may be when the anchor is displayed as part of a media item, or during the process of following a link. Style information for anchors is recorded in the presentation specification for an anchor in atomic and composite components, in channels and in the presentation specification of a link specifier (applicable to the source and destination anchors). Style information in the link specifier can be used for expressing how the source and destination anchors will be displayed when the link is followed.
- *Transition effect*
  Transition effects are the special effects which can be applied to the beginning or end of single events. These can be stored with the atomic component, with a channel associated with the atomic component, with an ancestor composite component, and with a link specifier (applicable to destination contexts). Transition effects can also be applied to the source and destination contexts when following a link. These are stored as the transition information of the link component.

The author requires to be able to assign styles for groups of objects and for overrides for single events. This gives the advantage of high-level operations, at the same time maintaining flexibility for overrides. Styles for different groupings should be able to be assigned via channels and composite components. Styles for single events should be able to be assigned via atomic components.

Each style type should be selectable from its own resource of styles. A media item style may be applicable to one or more media types. The anchor and transi-

tion styles, however, should be applicable to all media types. Where the same literal interpretation is not possible (for example for audio and visual media) there should be different interpretations for all media types. This allows an author to assign a style to any component without needing to know whether it is applicable.

Discussion

We advocate the use of style resources, because the more information that is included in resources, the better it is for style integrity, reuse and maintenance. These increase consistency within a presentation and facilitate applying changes at a high level. We are not aware of any system that currently allows the specification of styles for multimedia, or that allows styles to be assigned to channels or throughout a composite structure. There exist already, however, styles for paper documents and hypertext, in particular DSSSL [BuRL91], CSS1 [LiBo96] and XSL [ABCC97]. These are commonly used in text-based environments for specifying structured documents[4]. The application of styles to hypermedia can be very powerful, but also complex, as shown in initial work being carried out on style sheets for hypermedia, [OHRE97].

5.5.3 Channels

A channel collects together data type, layout and style information into one object which can be re-used by several atomic components. It can also have semantic attributes associated with it. Having defined a channel it can be re-used within a document and in multiple documents. The channel is used at playback time for resource control, but we do not discuss this further here. Where a document model uses channels, an atomic component requires a channel to be assigned to it. A channel itself also requires an editing environment.

Edit channel

While the authoring environment needs to know the precise data format for a media item, it is easier for an author to use media types. In other words, the author need only know that they are using an image or a video, and not know the actual data format used. The system should be able to work this out from the data file header information.

The position of a channel needs to be defined in terms of some base layout, for example the screen, a window or another channel.

The styles appropriate to the channel are those applicable to the atomic component: namely media item style, anchor style and transition effects. These should be selectable from the same resource as for the components.

The attributes of a channel should be selectable from an attribute resource, otherwise the semantic relationships among the attributes are unknown.

---

4. In particular documents using SGML style mark-up. CSS1 is the style sheet language for HTML. XSL is a simplified version of DSSSL for use on the World Wide Web.

Assign Channel
In the atomic component editor the channel associated with the component needs to be selectable from a list of appropriate channels. For example, an atomic component with an ASCII file as a media item could be assigned a text or an HTML channel. If the author has already assigned a media item to the component then the system should show only those channels appropriate for the data type. The system should show only those channels not already assigned to another atomic component playing at the same time in the same composite. The author should also be able to select a current layout.

Discussion
The channel construct allows a number of properties and resources to be allocated to groups of events. Channels themselves can also be grouped and inherit properties from the group, but use overrides as appropriate.

An author should be able to see groups of channels combined together in a layout, that is a grouping which the author perceives as useful. It is likely that a layout fills a window. Note that the composite component hierarchy is orthogonal to the layout channel hierarchy associated with the descendant atomic components. Channels or layouts belonging to a layout channel would inherit styles and transition effects from their parent layout. We advocate the use of layout channels in an authoring environment.

From an author's perspective, it is a restriction that channels have a fixed size and position. Channels are not only introduced as a high-level style/layout specification, but, at least as importantly as virtual resources that a playback environment can use for pre-calculating screen usage, or audio channel allocation. While from an author's perspective it is useful to vary the size or position of the channel it is questionable whether this is still acceptable for the channel's virtual resource function.

In summary, the introduction of channels into a hypermedia presentation environment brings with it advantages and disadvantages. The advantages are in the area of applying and manipulating resources at a high level. The disadvantages are in the area of flexibility of layout specification. We perceive the advantages as outweighing the disadvantages, thus advocate the use of channels in an authoring and in particular the playback environment.

5.5.4 Attributes

Attributes allow semantic information to be attached directly to different parts of a hypermedia presentation. Having assigned these they can be used for media-independent information retrieval and more automatic link creation. From an authoring perspective, we are concerned with assigning attributes to the appropriate components and not editing the attributes, nor their semantics, as such. What is important is the level of detail at which attributes can be assigned. Attributes can be associated with an atomic component, each anchor

of an atomic component, a link component, a composite component, each anchor of a composite component and channels.

An author should be able to assign one or more existing attributes to a particular component, or an anchor within a component. The author should be allowed to add to the list of attributes, but only in a way that can be re-used by other applications. Ideally a large thesaurus should be available, allowing searches, generalisations and specializations of concepts.

Discussion

The authoring systems discussed in chapter 4 make no mention of attributes, with the exception of the Athena Muse, [HoSA89], which gives no details on how the links among composite structures are based on high-level abstractions. The exact use of attributes, and in particular how they can be used for information retrieval and more automated generation of links, is currently a topic of research. We have carried out some initial work in this direction, see for example [HaBu95a], [WBHT97].

Current technology requires that each attribute be assigned personally by an author. This is, yet another, tedious task that we should try to automate as far as possible. Tools for, for example, showing salient stills of a video could be used for an author to point at objects and state (preferably verbally to avoid long point and click sessions in menus) what they are. The system should be able to find the boundaries of the object, and assign the same attributes whenever the object reappears later, or whenever a similar object appears (requesting confirmation from the author when this is ambiguous). Ideally, internal consistency checking should also be going on, so that the system can warn the author if different attributes are assigned to multiple occurrences of the same object.

The use of attributes within documents is not necessary for creating a hypermedia presentation. We believe, however, that with the increased use of internet technology the demand for finding objects of different media will increase greatly, with the consequence that assigning attributes, initially to objects and later to presentations, will become a standard feature. An authoring environment should thus provide standard facilities for choosing and assigning attributes.

5.6 Summary and Conclusion

In this section we go through the four layers of our authoring model and restate the most important conclusions. This results in a list of requirements for a next generation hypermedia authoring system.

The two main, and conflicting, goals of building an authoring system are that it is comprehensive in the data structures it is able to edit, and that the authoring effort is reduced as much as possible. The former goal prescribes a minimal functionality that has to be supported, and the latter requires an exploration of

the authoring interfaces for providing the functionality. A corollary of reducing the authoring effort is that a presentation should be authored only once, but contain sufficient information for playback on differing end-user platforms.

We divide the authoring environment into four separate but communicating layers—data, component, document and resource. This facilitates maintenance and flexibility of associating different resources with the same components. The data layer shields the other layers from data-dependencies, and the document layer supports alternative views of the component layer. These different views supply the information needed for a particular task, thus focusing the attention of the author.

5.6.1 Data layer

The separation of the data layer ensures that data dependencies are hidden from the main authoring process, allowing the presentation of a uniform interface regardless of data dependent details.

Media item

While we advocate the separation of the media item and its containing atomic component, and in so doing appear to lessen the importance of a media item, it remains the basis of a presentation. The author requires tools for selecting one media item from many, and then to select the appropriate part of the chosen media item for inclusion in the presentation. The latter requires the provision of media-dependent tools for selecting part of a media item. The environment also needs access to descriptions of the data formats of the media items, which we recommend be stored as a format resource.

5.6.2 Component layer

The component layer records the specifications for a hypermedia presentation by means of the atomic, composite and link components.

Atomic

The main benefit of introducing an atomic component is that it hides the data-dependencies of the media item from the rest of the environment. This allows complex presentations to be constructed while retaining maintainability.

To increase consistency and enhance reuse of resources, the style and attributes of an atomic component should be selectable from separate resources.

While an atomic component contains information relating to a particular media item, it is still possible to exchange equivalent items, where this does not make other properties invalid. This allows media item selection to be made at runtime, depending on the available display and transport resources. This facilitates the creation of platform independent presentations.

Composite

From an author's perspective, atomic and composite components can be treated in the same way, thus simplifying the authoring process. Composition facilitates

the creation of complex presentations from other smaller presentations, and also allows the reuse of a composite in different places in a larger presentation. The system abstractions should mirror the author's tasks as closely as possible in order to reduce the complexity of the authoring process. In particular, the narrative structure of the presentation can be reflected in the composition structure. This structure should be used as much as possible for generating other detailed aspects of the presentation, in particular the timing constraints.

Composition supports the specification of properties, such as styles and attributes, on a more global level thus relieving the author of repetitive work and enhancing style consistency.

The introduction of the composite anchor allows multiple anchors to be collected together in a structure that the author is able to perceive and manipulate as a whole. This allows link-ends to cross media boundaries thus better reflecting the semantics of the link.

Link

The main advantage of introducing a link component is that it allows the author to specify multiple routes for the end-user to follow through a number of multimedia presentations. Linking in hypermedia requires the specification of how the presentation will behave when moving from the source to the destination of the link. This requires the specification of source and destination anchors, components and contexts, along with transition information. The transition information includes the duration of the link.

The styles and attributes applicable to a link component should be selectable from separate resources to enhance reuse of style and attribute resources.

Rather than requiring an author to create links on an individual basis, a higher level approach can be achieved by creating links among abstractions instead of among concrete components.

5.6.3 Document layer

While the component layer is able to record all aspects of a presentation, editing individual components is rarely the best view for constructing a presentation. The impression of the presentation as perceived by the reader is dominated by such aspects as timing and spatial layout, hence they are sufficiently important to warrant specialist authoring support. Link management support is needed for dealing with the otherwise intangible collection of links.

Timing

At the document level, timing is a crucial aspect in authoring a hypermedia presentation, so that the author should have access to an overview of the timing. Timing should not, however, be used as the basis for authoring, but follow from the composition structure where possible. Deriving the start time and duration of an event from the structure saves the author work through not having to specify the details for every event.

When the start times of an event cannot be derived from the structure, it should be specified using timing constraints between events rather than absolute start times. This reduces the authoring burden when making changes to the presentation. Constraints should also be specifiable between groups of events. To facilitate the editing of constraints between groups of events, the composition structure and time should be displayed in the same view. Authoring tools for specifying timing constraints at a level above that of single synchronization arcs should be provided where possible.

As well as the duration of the atomic and composite components, the author also needs to be able to specify the duration of links, that is the duration of the transition from the currently playing presentation to the destination of the link.

An example of specification of timing via constraints is implemented in Firefly [BuZe93]. An example of displaying the composition structure and time in the same view is given in [Acke94].

Spatial layout

Specifying the position of events in a multimedia presentation is, along with the timing, one of the most fundamental requirements. An author should be able to view the positions of co-occurring events while editing the position of an event. We recommend that position and size be specified with respect to a channel, and not with respect to another event, which facilitates consistency of layout. In most authoring systems, spatial layout is not specified via channels, but by assigning a position on the screen to the media item. This is done by positioning the item where it should appear, [ENKY94], [HTOH96], or by specifying the x and y positions over time, [Acke94]. In both cases it is difficult to get an overview of the position of the object relative to other objects or over time. A timeline view provides the author with easy access to a screen view from any point along the timeline, e.g. Director, [West93].

The position of a component can be a function of time, so that the associated media item traces out a path. We advocate the provision of a visualisation for an event's path, but are unable to recommend a particular approach. Path specification is illustrated in [West93] and [Acke94]. As well as allowing the position of an object to change with time, the size of an object should also be able to change with time. Size changes over time can also apply to a transition. Neither of these is, as far as we are aware, already implemented in a multimedia authoring system.

Link management

A hypermedia author requires to specify, verify and maintain links among separate presentations. The author needs to check that the possible paths through the presentation are meaningful to the end-user, whether these paths are complete or still under construction, and whether the transitions are appropriate. A link management system should aid the author in these tasks, as a minimum by pro-

viding author-specifi able lists of links which can be used as check-lists. A number of aspects of link management are illustrated in Microcosm, [HaDH96].

Presentation Control
Direct communication between the editing environment and the playback system should be possible during the authoring process. The editing environment should be able to pass any part of the presentation's specifi cation to the playback system, and the playback system should be able to communicate to the authoring environment which parts of the presentation are actually playing.

5.6.4 Resource

The use of resources allows properties, such as data format, style and attributes, to be assigned on a re-usable basis and leaves them open to external standardisation. Examples of data formats are GIF, PNG [Bout96] and MPEG [Pere96], of styles CSS1 [LiBo96] and of attributes PICS [KMRT96]. Channels are a useful resource abstraction and can themselves be allocated properties from the other resources.

Data format
External information in order to be able to interpret the data format used by a media item should be available via a resource.

Style
The use of styles in hypermedia is not yet commonplace, although their application is a powerful tool allowing different styles to be applied to the same fi nal document. Without the use of a style resource there is a lack of style consistency, hence we advocate the use of a style resource. Styles allow the appearance of a presentation to be changed with very little effort. Style information applicable to hypermedia includes media item, anchor and transition effect styles. These are three separate style resources media item, anchor and transition, each of which is applicable to the three component types.

Channel
The channel construct allows a number of other resources to be allocated to groups of events. Channels themselves can also be combined together in a layout. Channels, or layouts, inherit their resources and position from their parent layout. Any or all of these should be overrideable in a particular channel instantiation.

There is a tension in channels being used as resource allocators and their potential infl exibility in layout specifi cation. We currently hold the view that, for the case of presentations destined to be played on multiple platform types over a network, the resource arguments override those for layout fl exibility

Attributes
The use of attributes within documents is not necessary for creating a hypermedia presentation. The exact use of attributes, and in particular how they can be used for information retrieval and more automated generation of links, is cur-

rently a topic of research. We believe that the demand for finding objects of different media will increase greatly, with the consequence that assigning attributes will become a standard feature. This will only be useful if the attributes are held in a separate resource.

5.6.5 Designing a real system

This chapter describes the editing facilities required for a complete hypermedia authoring system. These on their own are insufficient for a useful design of a system, since some of the requirements may conflict. The following chapter describes the authoring system CMIFed which incorporates a large number of the listed features in a unified set of tools.

Appendix

Model elements			References to sections
Channel	Presentation Specifi cation	Channel ref. Position & extent Style	5.5.3 Channels 5.4.2 Spatial layout 5.5.2 Styles (media item, anchor, transition effect)
	Attributes		5.5.4 Attributes
	Media type		5.5.1 Data format
Atomic Component	Presentation Specifi cation	Duration Channel ref. Position Extent Style	5.4.1 Timing (duration) 5.5.3 Channels 5.4.2 Spatial layout 5.4.2 Spatial layout 5.5.2 Styles (media item, anchor, transition effect)
	Attributes		5.5.4 Attributes
	Anchors	Anchor ID Pres. Spec. Attributes Value	Specifi ed 5.5.2 Styles (anchor) 5.5.4 Attributes 5.2.1 Media items, 5.4.1 Timing (duration, start-time) 5.4.2 Spatial layout (size, position w.r.t. content)
	Content	Media item ref. Data-dep. spec.	5.2.1 Media items 5.4.1 Timing (duration) 5.4.2 Spatial layout (size)
Temporal Composite Component	Presentation Specifi cation	Duration Sync. arcs Style	5.4.1 Timing (duration) 5.4.1 Timing (start-time) 5.5.2 Styles (media item, anchor, transition effect)
	Attributes		5.5.4 Attributes
	Anchors	Anchor ID Pres. Spec. Attributes List of anchors	Specifi ed 5.5.2 Styles (anchor) 5.5.4 Attributes 5.3.1 Atomic components, 5.3.2 Composite components
	Children	Comp. ref.	5.3.1 Atomic components, 5.3.2 Composite components
Atemporal Composite Component	Presentation Specifi cation	Initial activ. state Play/pause Style	5.3.2 Composite components 5.3.2 Composite components 5.5.2 Styles (media item, anchor, transition effect)
	Attributes		As for temporal composite
	Anchors		As for temporal composite
	Children		As for temporal composite

TABLE 5.1 AHM elements and where discussed in this chapter

Model elements			References to sections
Link Component	Presentation Specifi cation	Duration	5.4.1 Timing (transition duration)
		Relative position	5.4.2 Spatial layout
		Style	5.5.2 Styles (link)
	Attributes		5.5.4 Attributes
	Anchors		Not discussed
	Specifi ers	Source cont. act.	5.3.3 Link components
		Dest. cont. p.p.	5.3.3 Link components
		Style	5.5.2 Styles (media item, anchor, transition effect)
		Anchor	5.3.1 Atomic components, 5.3.2 Composite components
		Context	5.3.1 Atomic components, 5.3.2 Composite components
		Direction	5.3.3 Link components

TABLE 5.1 AHM elements and where discussed in this chapter

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

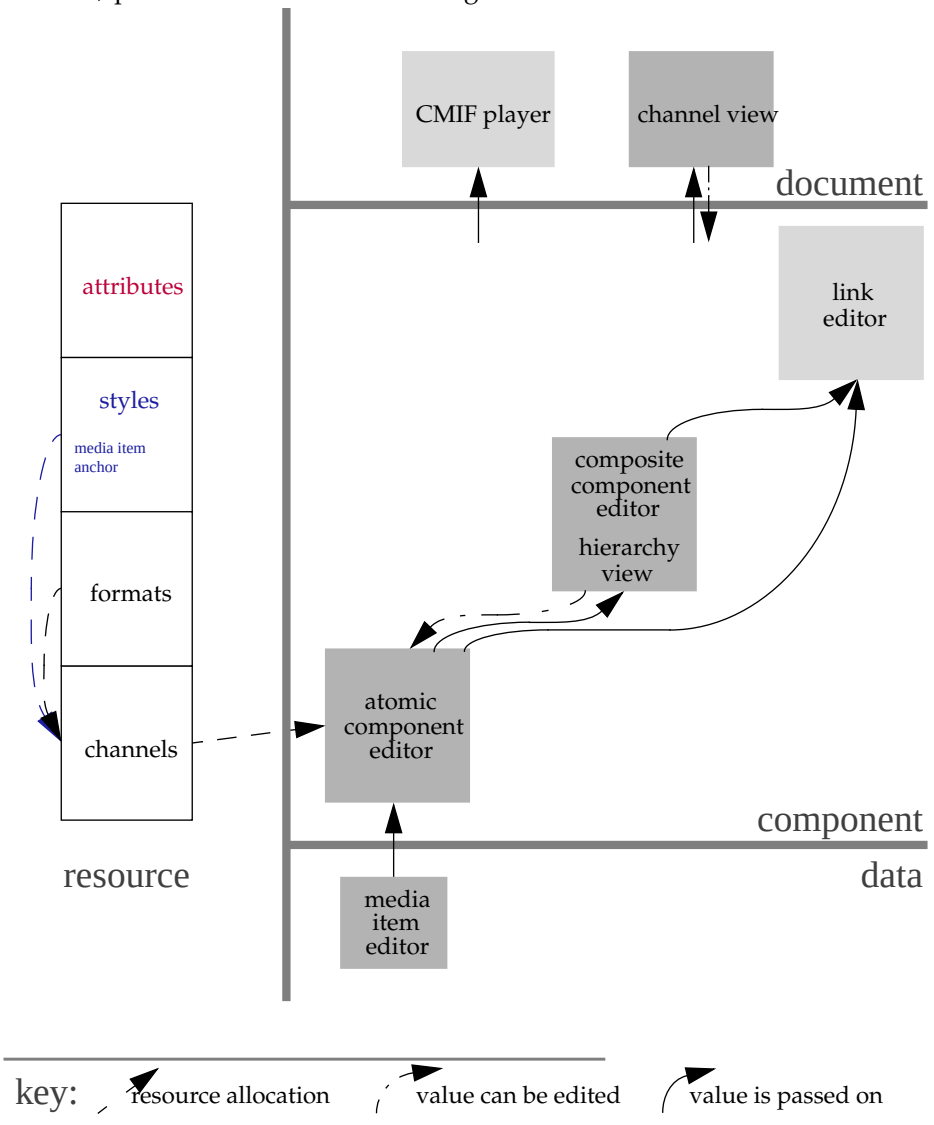


Figure 6.1. Overview of data fl ow for four layers

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

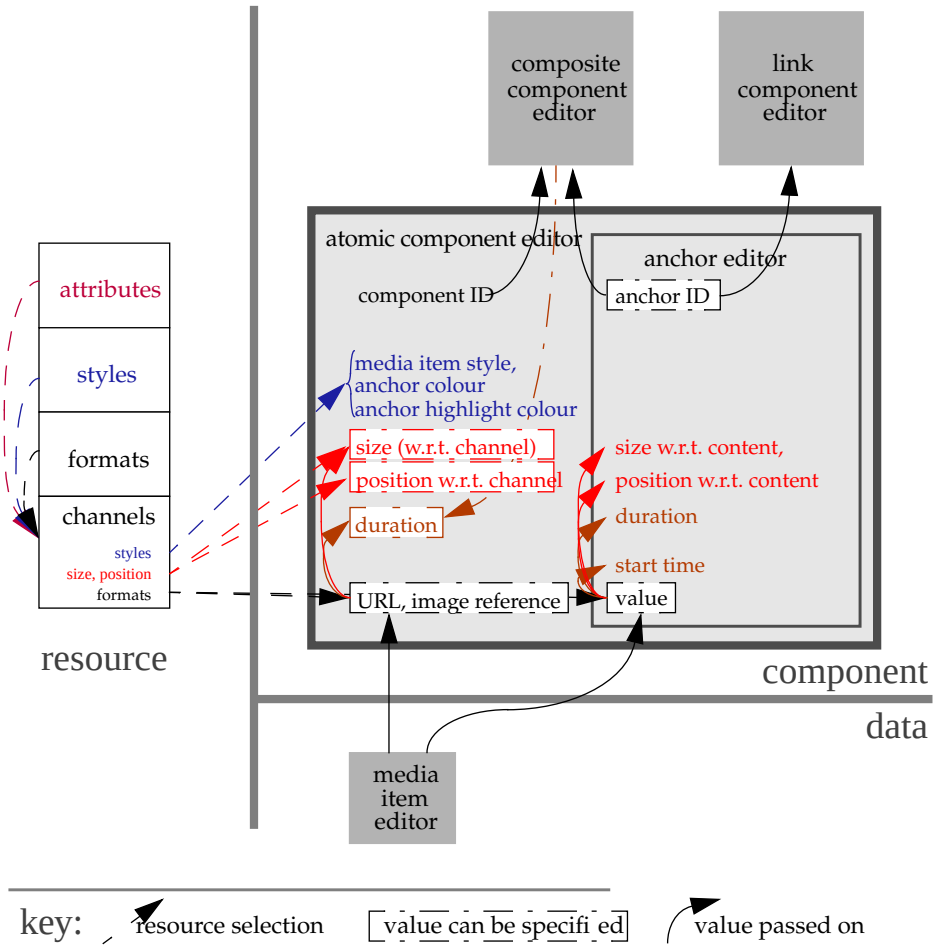


Figure 6.2. Data flow in and out of atomic component editor

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

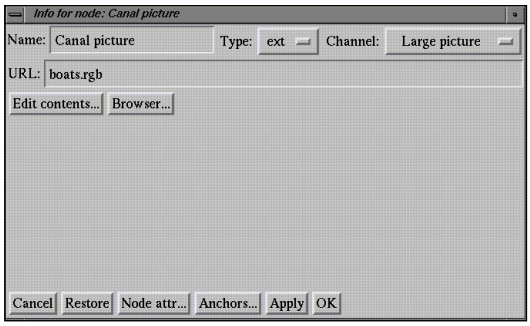
Attributes and anchor-specific presentation information are not supported within CMIFed. The consequence of the latter is that all anchors within the same media item have the same presentation style. This affords consistency, but does not allow, for example, the distinctive highlighting of individual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

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

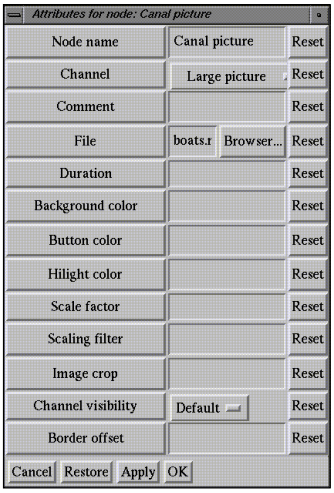
- A *normal anchor* can be used as the source and/or destination of a link, and is specified as described above.
- An *auto-firing anchor* is one that triggers the following of the associated link automatically when it is displayed.
- A *destination only anchor* can be used only as the destination of 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

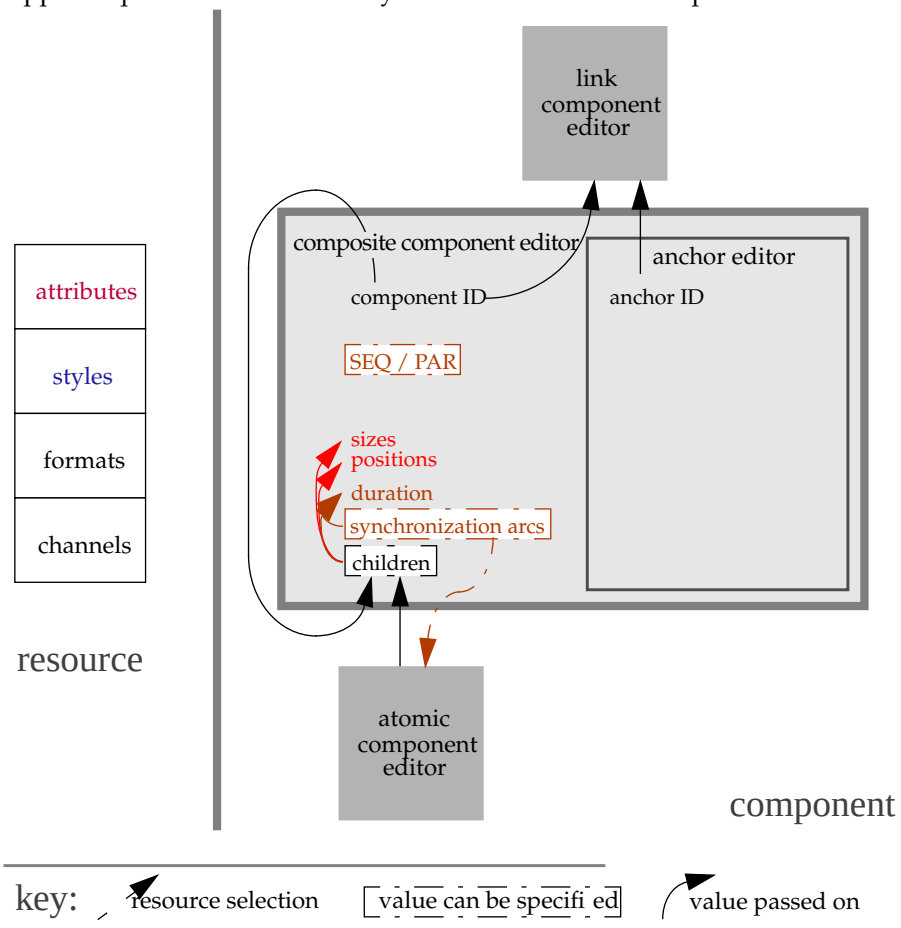


Figure 6.4. Data fl ow in and out of temporal composite component editor

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-

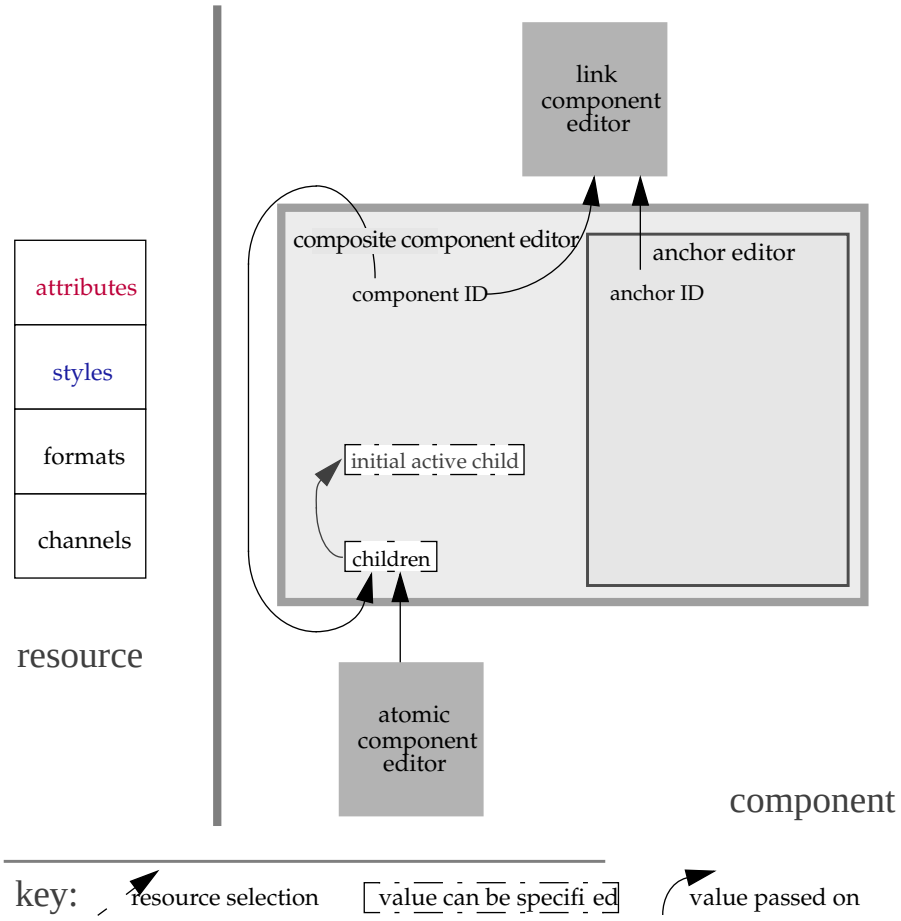
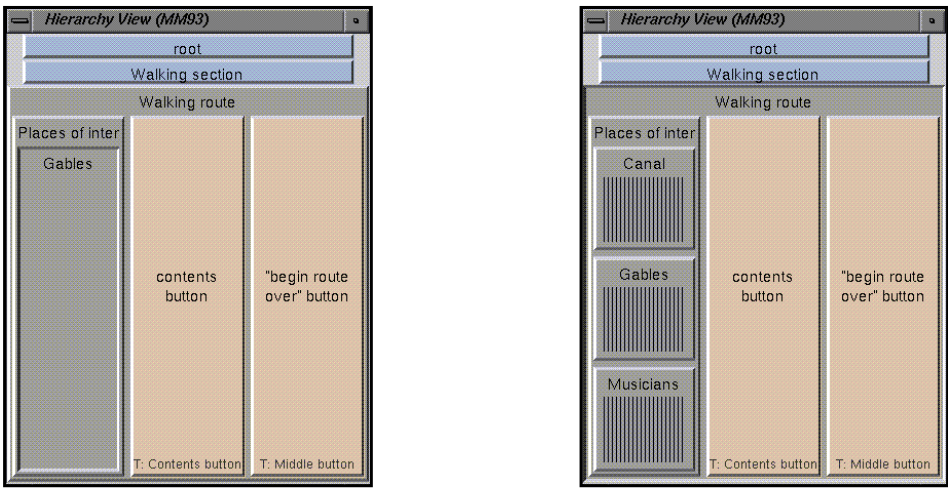


Figure 6.5. Data fl ow in and out of atemporal composite component editor



(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

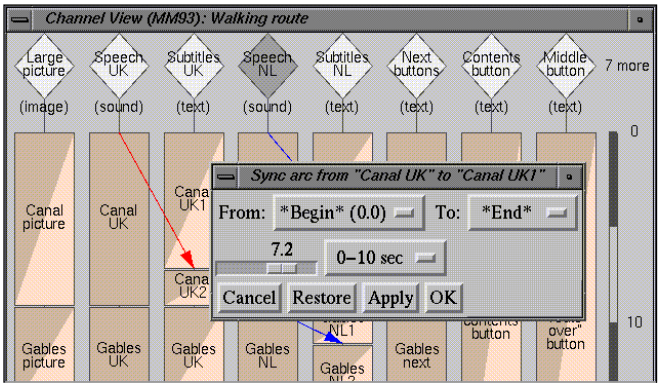


Figure 6.8. Creating a synchronization arc

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

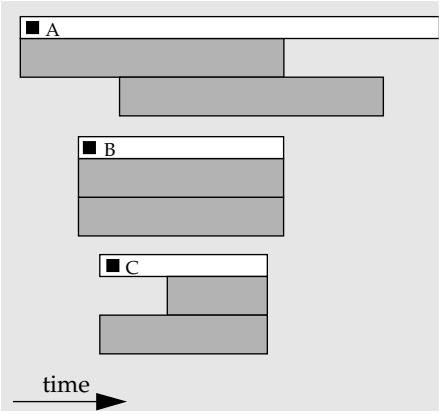
parent (A). In order to show structure and time unambiguously in one view three dimensions are needed.

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:

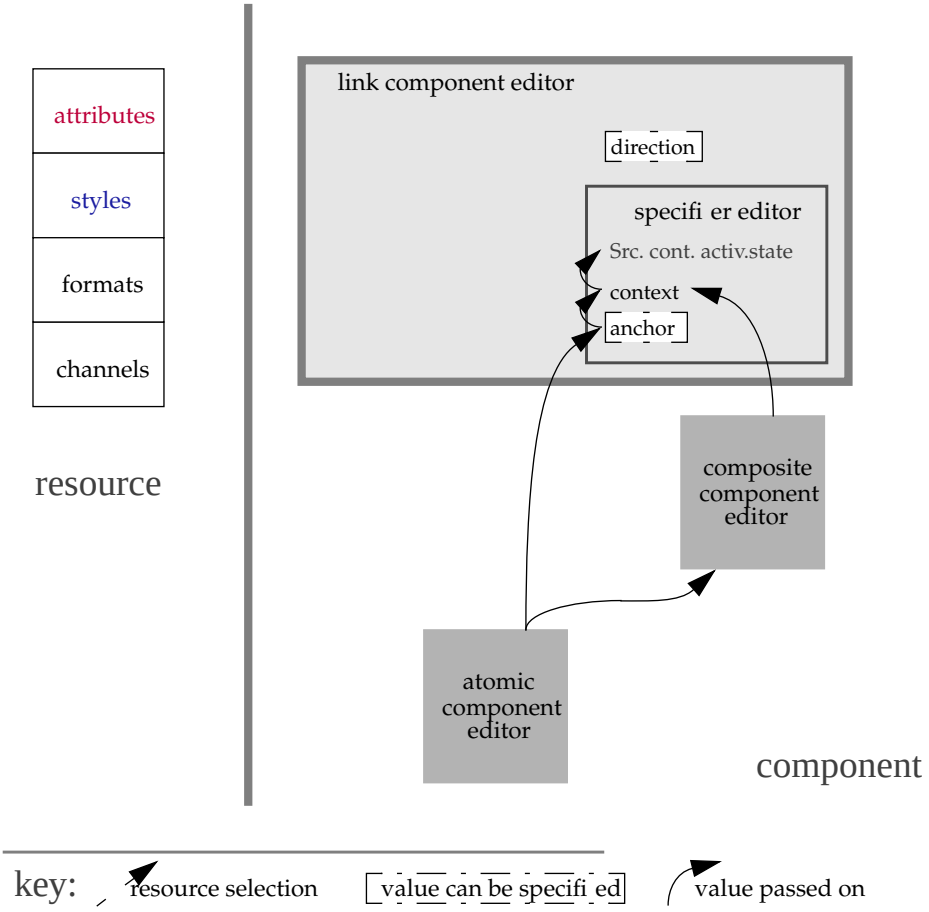
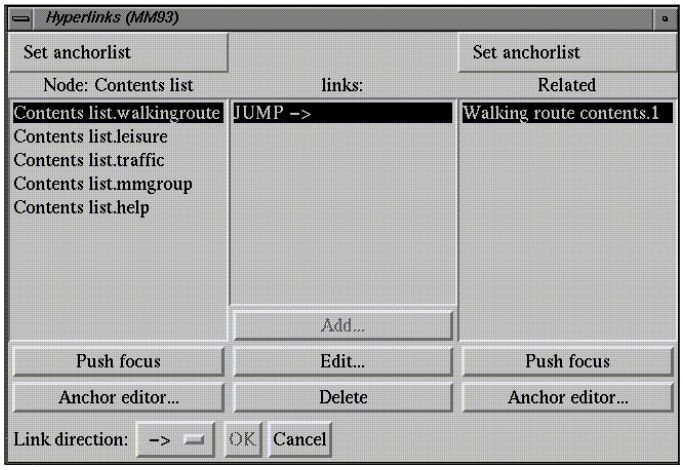


Figure 6.10. Data flow in and out of link component editor

- *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.

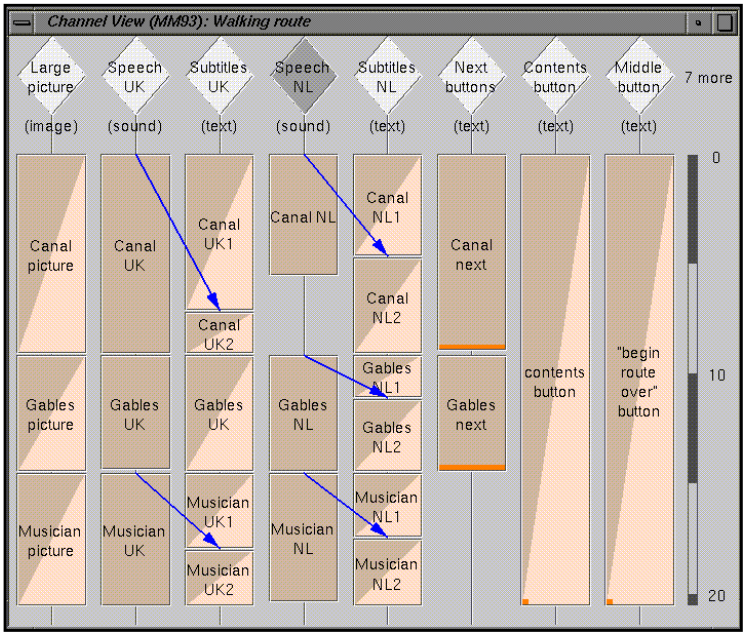
6.4.1 Timing

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.

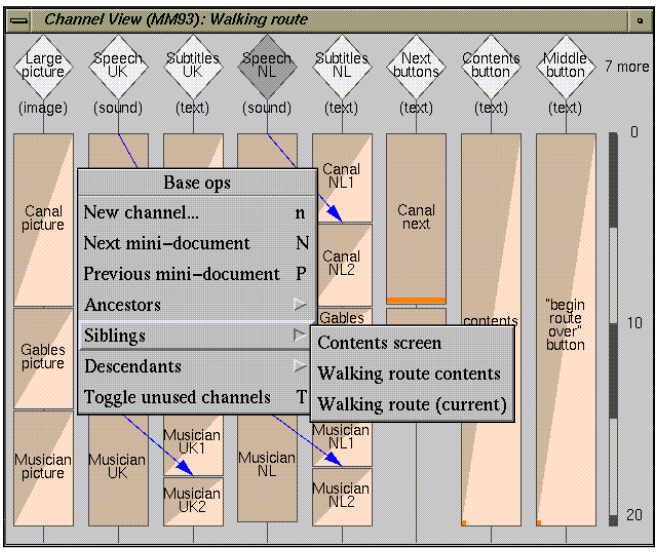
Duration of a link transition
CMIFed does not support the specification of a link transition duration, as discussed in the previous section.

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.

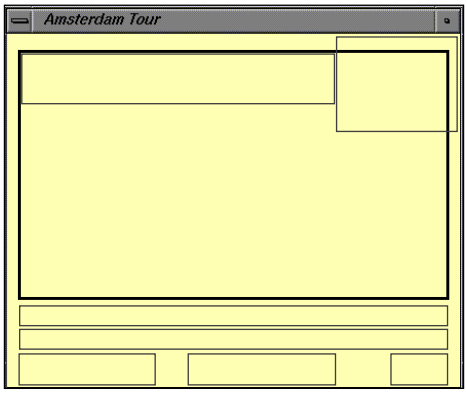
The extents and position of a channel are defi ned with respect to a layout channel. An author is able to edit a channel's size and position while viewing other channel extents in the same layout, Fig. 6.14.

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

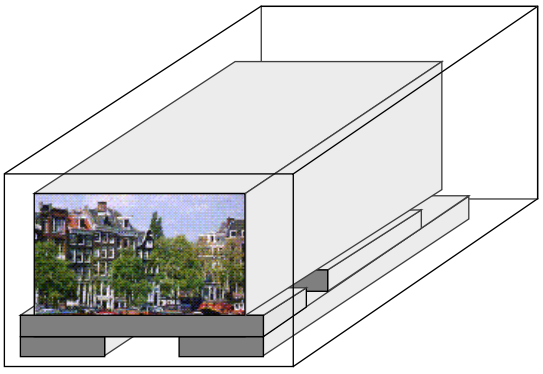


Figure 6.15. 3D layout view

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

7 Summary and Conclusions

In this chapter we first give a summary of the conclusions reached throughout the body of the thesis. We then discuss two applications of our work, the first based on the model defined in the thesis and the second on the authoring system described.

7.1 Summary

This thesis derives the requirements and defines a document model for hypermedia. The model combines synchronization relations among multiple, possibly continuous, media items along with linking structures among the components of the document. The thesis goes on to specify authoring system requirements for the model and describes the implementation of the authoring system CMIFed.

7.1.1 Model

In Chapter 2 we derived the requirements for a hypermedia model. We argued that a model sufficiently expressive to describe a hypermedia presentation is required to include the following:

- multiple media types and data formats,
- temporal and spatial layout information,
- grouping of individual and group elements,
- the ability to address part of an individual media item,
- relationships among groups or individual elements, and
- media-independent descriptions of media items.

Given the importance of temporal and spatial layout for a multimedia presentation, we described these requirements in greater detail. We also described the requirements for a document model for describing link activation, since for presentations containing multiple synchronized, continuous elements a precise way of defining the activation state of the presentation is required. The reader should also be able to control the playing of the presentation itself at run-time.

We concluded that existing hypertext and multimedia models were insufficient as models of hypermedia. Of the requirements for a hypermedia document model, those that are not already satisfied by existing models are:

- context for linkends, required because of the multiple components collected together in the source and destination contexts;
- transition information for a link, since on traversing a link the presentation should remain a continuous presentation as perceived by the reader;
- attributes for anchors, since these portray basic real-world objects, in contrast with a media item which is a basic system object.

In Chapter 3 we defined the Amsterdam Hypermedia Model (AHM) which satisfies the requirements derived in Chapter 2. In order to maintain an overview of the parts of the model, it is divided into the three Dexter layers: within-component, storage, and runtime. The storage layer communicates with the within-component layer by means of an anchoring mechanism which encapsulates data-dependent details in the within-component layer. The runtime layer communicates with the storage layer by means of presentation specifications which allow the styles of components to be stored independently of the components themselves.

The AHM incorporates the following novel extensions to the Dexter hypertext reference and the CMIF multimedia models:

- The presentation specifications within the model are explicitly stated as temporal, spatial, style and activation information. Each aspect occurs throughout the model and we demonstrated how the occurrences relate to one another. This allows all aspects to be edited for a single component, or the same aspect for multiple components.
- Media item reference, anchor reference and channel reference, in addition to a component reference, are used throughout the model. This allows components to be selected on the basis of, for example, semantic annotations.
- Content is specified explicitly as a media item reference along with a corresponding data-dependent specification, which allows different parts of a single media item to be included in multiple presentations.
- Anchors have been extended to include semantic attributes and presentation specifications, including start time and duration for an atomic anchor of a non-continuous media type. The semantic attributes allow anchors to be searchable. The presentations specifications allow different appearances to be assigned to anchors and in non-continuous media types the anchor value may be visible for only part of the duration of the component.
- Composition of anchors was introduced to allow the collection of items of similar semantics in different media, thus reducing the number of links required.
- Composition of components is of two types: temporal and atemporal. Dexter expressed only atemporal and CMIF expressed only temporal. The inclusion of both types of composition within one model allows the composition of presentations where temporal relationships are known only at runtime.

- Activation state information has been incorporated throughout the model. This includes initial activation state, play/pause state and change in activation state on following a link. Activation state information allows the initial activation state of the presentation to be recorded in the document and specifies how this changes when a user follows a link.
- Link components have been extended to include context and activation state in the link specifier. The context allows the specification of how much of the running presentation is affected on following a link. The activation state specifies the behaviour of the source and destination contexts.
- Transition information, including transition duration and special effect, has been incorporated in the model. This allows the specification of the behaviour of the presentation when a user follows a link.

While the parts of the model have been shown to be necessary for describing a hypermedia presentation they must also be shown to be sufficient. We demonstrated this by describing the models implicit in a selection of existing hypertext, multimedia and hypermedia systems in terms of the AHM.

We concluded that our objective of providing a comprehensive, yet not overly complex, model for hypermedia presentations has been satisfied by the AHM.

7.1.2 Authoring

To support the author in the creation of hypermedia documents an authoring environment is required.

Chapter 4 analysed the authoring paradigms used in a selection of existing multimedia authoring systems. The paradigms illustrate different ways of providing similar functionality in a hypermedia authoring environment. They do not, however, provide a solution to the problem of which functionality should be provided in such an authoring environment. The paradigms were analysed for their suitability for different parts of the authoring task, namely creating narrative structure, temporal information, spatial layout and links among individual presentations. Structure based systems are more suited to editing the structure of a presentation, and, where the structure reflects the temporal structure, are also useful for editing the presentation's timing. Timeline based systems are more suited to showing the timing throughout a presentation and the timing relationships among parts of a presentation. None of the paradigms discussed is particularly suitable for editing layout or for creating links, although the structure-based paradigm allows the different parts of the link to be specified.

We demonstrated how an event within a multimedia presentation can be described using each of the paradigms. We concluded that each paradigm is most suited to a particular editing task, that no single paradigm is sufficiently powerful for covering all editing aspects of a hypermedia presentation, and that several interfaces within a unified environment are required.

Chapter 5 derived the authoring requirements for a system that supports the creation of hypermedia documents conforming to the AHM. To maintain an

overview of the authoring environment it is divided into 4 layers: resource, data, component and document. The resource layer is a generalisation of the Dexter presentation specifications. The data layer corresponds to the within-component layer and the component layer to the storage layer. The document layer includes both a static analysis of the aspects stored in the component layer as well as run-time aspects.

The resource layer contains the resources used for the different aspects of the document, for example a data format resource required for interpreting the data, style information for fonts, semantic attribute information dependent on the application domain, and layout information. The importance of including the resources as a separate layer is that each can be replaced by a similar resource while leaving the document structure itself unchanged. This allows multiple presentations to be generated from the same underlying document structure, e.g., layout can be tailored to specific output environments.

The component layer is supported in an authoring environment by allowing the individual editing of the components themselves. An important aspect of the component layer is that atomic and composite components have been defined so that they can be treated equivalently. This facilitates a uniform approach to inserting and deleting components in a document and enhances maintainability.

Temporal and spatial layout play a particularly important role in hypermedia presentations, and these aspects have to be coordinated among multiple elements. The document layer allows the editing of these aspects and communicates the information which requires to be stored to the component and resource layers.

As well as stating the requirements for the document layer, illustrations of potential user interfaces were given for aspects such as editing temporal and spatial information. In particular, timeline illustrations were given for showing temporal constraints, changes in tempo, and navigating the presentation timeline.

Chapter 6 described the authoring environment CMIFed and stated where it deviates from satisfying the requirements stated in Chapter 5. This demonstrated that an authoring system broadly conforming to the requirements derived in Chapter 5 can be implemented. The omissions from the implemented system were categorised according to whether they would be worthy of implementation, or, from experience, were not found to be necessary.

Our conclusions in Chapter 6 were that the AHM is not an overly complex document model, since the key parts of the model have been implemented in a working system, and that the specified functionality derived in Chapter 5 is broadly implementable.

7.2 Application of work

The work reported in this thesis has contributed to and benefited from important international collaboration. The most notable of these we report below. The first is more closely related to the model part of the thesis, the second to the authoring part.

7.2.1 SMIL

The World Wide Web Consortium [W3C97] is an international industry and research consortium which coordinates the development of specifications and reference software that are made freely available. The goal of the consortium's Working Group on Synchronized Multimedia [SYMM97] is to define a declarative document format for synchronized multimedia documents for the Web. The name of the format is SMIL—Synchronized Multimedia Integration Language—pronounced "smile" [Hosc97b]. This language provided an opportunity to test the robustness of the AHM, since the goals for the documents it should be able to describe are very similar to those for the AHM, [Hosc97a].

The model proved to be a solid base from which to work and the AHM played a major role in defining the requirements for the SMIL document model. While it was the main influence on the definition of the requirements, for pragmatic reasons, such as simplicity of use and implementation and acceptance of the first version, some of the finer details of the AHM have had to be postponed until later versions of the SMIL language. In essence, however, the SMIL model contains the same basic parts as the AHM:

- references to the basic data items using URL's (a reference to part of a data item is not yet included);
- data items are encapsulated within atomic components;
- anchor specification is possible, but is less prominent than in the AHM;
- temporal composites, of types parallel and sequential, containing atomic, parallel or sequential elements;
- a repeat attribute which allows the content of an element (atomic, parallel or sequential) to be repeated (this is captured implicitly in the AHM as a specified duration in terms of the intrinsic duration of the component, and is a particular case of satisfying the specified duration by repeating the content);
- synchronization relations (but these can be created only among siblings to alleviate the complexities of the player software);
- layout is defined in a separate structure and referred to from an atomic component;
- links can be defined among atomic or composite components.

While the AHM is in general a superset of the model implicit in SMIL, a number of runtime aspects included within SMIL are outside the scope of the AHM:

- lip synchronization, that is synchronization defined at every point during the presentation of two or more synchronized continuous media elements;
- switching among different data types at runtime.

Major aspects included within the AHM but not in SMIL are:

- atemporal composition. A SMIL presentation (in version 1.0) has a single contiguous temporal extent. It is expected that this will be included in future versions to better support linking among presentations;
- links with multiple sources and destinations;
- the association of semantic attributes with components.

The AHM proved to be sufficiently expressive that it could be used as a basis for the SMIL language. The modelling aspects that were of particular relevance were the aspects which should be addressed in the language, in other words the requirements as given in chapter 2, and the details of link descriptions. The major obstacle in proposing the AHM as a base for SMIL was its complexity. The perceived needs of users do not currently match the capabilities of the technology, so that it was difficult to explain why parts of the model were required before the more basic parts were implemented. As a result, a number of aspects of the AHM are omitted in the first version. An illustrated high-level description of the model is given in [Bult97], and the current working draft in [Hosc97b].

7.2.2 Chameleon

The Chameleon project [Cham95] is an ESPRIT-IV research and development project whose aims are to define and implement multimedia authoring and presentation tools. One of the goals of the Chameleon project is to allow the creation of a single source document which can then be (semi-)automatically adapted for playback on a range of end-user platforms. As such, Chameleon is a suitable test-bed for the robustness of the underlying AHM model and for the authoring system CMIFed built around the model.

The original UNIX versions of the CMIFed authoring environment and player formed the initial core of the project and have been ported to the Macintosh and Windows platforms. The project thus had immediate access to an authoring environment which was able to create hypermedia presentations and whose functionality, broadly speaking, satisfied the authoring requirements as stated in Chapter 5 of this thesis. In addition to the availability of the authoring environment, CMIFed has been able to contribute to the Chameleon project through the document format is supports. The document format is based on the AHM and the information in the document format is recorded explicitly, which allows it to be translated with relative ease to other formats. Target formats currently being implemented in the Chameleon project are MHEG-5 [ISO97a] and SMIL.

The Chameleon project has contributed to the development of CMIFed with regard to our analysis of the authoring environment's user interface. Within the thesis, we have been able to give analysis of the user interface based on our own experience. The Chameleon project partners, who are already experienced in cre-

ating multimedia presentations using other tools, have been using the early versions of the system and giving feedback about the design of the user interface. This work is still in progress.

The main contributions of the CMIFed environment is that it supports a rich hypermedia document model. As a result, the complete environment has been able to be tailored to the SMIL document format with comparatively little implementation effort.

7.3 Discussion and future work

7.3.1 Extensions to the AHM

The AHM provides a sufficient model for hypermedia and forms a robust and well-formed model. The categorisations within the model of components, spatial layout, time, semantics, styles and linking behaviour would not need to be changed. The model could, however, be improved or extended within each of the different aspects, e.g. by separating the spatial layout hierarchy from the channel element, or by including a timeline object within a temporal composite.

The model could be extended further to include more runtime aspects. For example, by including a way of selecting among synchronized streams of information, or auto-filtering of links.

An extension to the model which may require changes beyond the detailed level is the inclusion of absolute time.

7.3.2 Facilitating authoring

To alleviate the author of tedious, and time-consuming, work, we have been investigating the potential of automating the authoring process by generating presentations from higher-level semantic descriptions. Initial work in this direction has already been carried out on two aspects. The first, reported in [WBHT97], is on the design of a system that integrates a store of media items annotated with semantic attributes with a means of making selections from them and combining them into a presentation that can be displayed to an end-user. The second is on the integration of a standard reference model for the process of creating dynamically generated multimedia presentations with a document model for hypermedia, in this case the AHM [HaWB98]. Further work needs to be carried out on the problem of relating pieces of content with semantic descriptions of the content, either semi-automatically or by hand.

A different approach is being taken for a similar goal, that of generating presentations suitable for different platforms, which relates to work being undertaken in the Chameleon project. The approach is to encode the CMIF format in HyTime [ISO97b], thus making the hypermedia semantics of the document format explicit. In order to express runtime aspects of a hypermedia presentation, extensions to HyTime are required [ROHB97b]. Given these, a hypermedia document expressed in HyTime can be converted to other document types, or to par-

ticular output formats, using standard tools. These include the declarative specifi cations of transformations using style sheets. Existing style sheets are, however, text based so that work needs to be carried out to extend them to hypermedia [OHRE97]. This is again a place where the AHM is applicable, since it makes explicit the aspects of a hypermedia presentation that need to be expressed in a style sheet. Work on a suitable style sheet document model is in progress.

7.3.3 Runtime aspects of a presentation

Aspects of a hypermedia presentation which the AHM does not address tend to be related to runtime issues and in particular issues related to the behaviour of large continuous media items being transported over networks with fluctuating available bandwidths. A number of aspects are already addressed in passing in the model, such as providing alternate data types for transmission based on user preference or available network bandwidth. Others, such as guaranteeing streaming of high-bandwidth video, are not. These require a solution at the level of the network protocol, such as guaranteeing network bandwidth available to an application. Work is being carried out on developing appropriate protocols, such as RTP [RTP95], RSVP [RSVP96], RTSP [RTSP96].

7.4 In conclusion

Information exchange has been a human activity for tens of thousands of years. The technologies used for recording and presenting 'documents' have developed from cave walls through papyrus and paper to computers. Computers make it possible to construct a complete processing environment for authoring, storage, play-back on different devices and document re-use. This is facilitated by a common underlying document model which separates document structure from presentation.

Our contribution has been to unify existing models of hypertext and multimedia into a richer hypermedia model and to improve computer support for these documents by stating the requirements for an authoring system and demonstrating the feasibility of implementation.

The work reported in this thesis is, however, only a part of a complete environment for the creation, storage, manipulation, transmission and play-back of hypermedia documents. Additional requirements are a generalised approach to processing documents, an agreed upon presentation model and a methodological approach to creating sets of hypermedia documents. The challenge is to ensure that these technologies will be as durable as the cave paintings of our ancestors.

Bibliography

[Acke94] P. Ackermann (1994). Direct Manipulation of Temporal Structures in a Multimedia Application Framework. In Proceedings: *Multimedia '94*, San Francisco, CA, Oct, 51 - 58.

[ABCC97] S. Adler, A. Berglund, J. Clark, I. Cseri, P. Grosso, J. Marsh, G. Nicol, J. Paoli, D. Schach, H.S. Thompson, C. Wilson (1997). A Proposal for XSL. `http://www.w3.org/TR/NOTE-XSL-970910`

[Adob90] Adobe Systems Incorporated (1990). Postscript® Language Reference Manual, second edition. Addison Wesley.

[Aimt97] Aimtech. IconAuthor version 7. `http://www.aimtech.com/`

[Alle83] J.F. Allen (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26 (11), Nov, 832- 843

[Andr96] E. André, J. Müller and T. Rist (1996). WIP/PPP: Knowledge-Based Methods for Fully Automated Multimedia Authoring. In proceedings EUROMEDIA '96, London, UK, 95 - 102.
`http://www.dfki.uni-sb.de:80/~andre/ed.ps.gz`

[Aron91] B. Arons (1991). Hyperspeech: Navigating in Speech-Only Hypermedia. In Proceedings: *ACM Hypertext '91*, San Antonio, TX, Dec 15-18, 133 - 146.

[BRFS96] R. Baecker, A.J. Rosenthal, N. Friedlander, E. Smith and A. Cohen (1996). A Multimedia System for Authoring Motion Pictures. In Proceedings: ACM *Multimedia '96*, Boston MA, Nov, 31 - 42.

[BFMR97] M. Bordegoni, G. Faconti, M.T. Maybury, T. Rist, S. Ruggieri, P. Trahanias and M. Wilson (1998). A Standard Reference Model for Intelligent Multimedia Presentation Systems. To appear in *Computer Standards and Interfaces*.

[Bord92] M. Bordegoni (1992). Multimedia in Views. CWI Report CS-R9263, December 1992. `http://www.cwi.nl/ftp/CWIreports/AA/CS-R9263.ps.Z`

[Bout96] Thomas Boutell (ed.) (1996). PNG (Portable Network Graphics) Specification, Version 1.0. W3C Recommendation 01-October-1996. REC-png.html. `http://www.w3.org/TR/png.html`

[BuZe93a] M.C. Buchanan and P.T. Zellweger (1993). Automatically Generating Consistent Schedules for Multimedia Documents. *Multimedia Systems*. 1:55-67.

[BuZe93b] M.C. Buchanan and P.T. Zellweger (1993). Automatic temporal layout mechanisms. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug, 341-350.

Bibliography

[Bufo94] J.F. Koegel Buford (ed.) (1994). Multimedia Systems. Addison-Wesley, New York, New York. ISBN 0-201-53258-1.

[BuHe93] J.F. Koegel (Buford) and J.M. Heines (1993). Improving Visual Programming Languages for Multimedia Authoring, *ED-MEDIA '93*, World Conference on Educational Multimedia and Hypermedia, Charlottsville, Virginia, June, 286 - 293.

[Bult97] D.C.A. Bulterman (1997). Models, Media and Motion: Using the Web to Support Multimedia Documents. In MMM '97, the International Conference on Multi-Media Modelling, Singapore, November 16-19.

[Pere96] D.C.A. Bulterman, G. van Rossum and R. van Liere (1991). A Structure for Transportable, Dynamic Multimedia Documents. In Proceedings of the Summer *USENIX* Conference, Nashville, Tennessee, 137-155.

[Burb92] S. Burbeck (1987, 1992). Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC). `http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html`

[BuKW94] V. A. Burrill, T. Kirste and J.M. Weiss (1994. Time-varying sensitive regions in dynamic multimedia objects: a pragmatic approach to content-based retrieval from video. *Information and Software Technology* Journal Special Issue on Multimedia 36(4), Butterworth-Heinemann, April, 213 - 224.

[Cail95] R.Cailliau (1995). A Little History of the World Wide Web. `http://www.w3.org/History.html`

[Cham95] CHAMELEON: An Authoring Environment for Adaptive Multimedia Documents (1995). ESPRIT-IV Project 20597. `http://www.cwi.nl/Chameleon/`

[Clot95] J. Clottes (1995). An Extraordinary Archaeological Find: A Decorated Paleolithic Cave in the Ardèche Region of France. `http://www.culture.fr/culture/arcnat/chauvet/en/gvpda-d.htm`

[Davi93] M. Davis (1993). Media Streams: An Iconic Language for Video Annotation. Telektronikk 5.93: Cyberspace Volume 89 (4) (1993) 49 - 71, Norwegian Telecom Research, ISSN 0085-7130 `http://www.nta.no/telektronikk/5.93.dir/Davis_M.html`.

[EnEn68] D.C. Engelbart and W.K. English (1968). A research center for augmenting human intellect. AFIPS Conference Proceedings 33(1) 395-410.

[Erfl93] R. Erfle (1993). Specification of Temporal Constraints in Multimedia Documents using HyTime. *Electronic Publishing*. 6(4), 397-411.

[ENKY94] S. Eun, E.S. No, H.C. Kim, H. Yoon and S.R. Maeng (1994). Eventor: An Authoring System for Interactive Multimedia Applications. *Multimedia Systems* 2: 129 - 140.

[FHHD90] A.M. Fountain, W. Hall, I. Heath and H.C. Davis (1990). MICROCOSM: An Open Model for Hypermedia With Dynamic Linking. In proceedings: *ECHT '90* (First European Conference on Hypertext), Nov, INRIA France, 298 - 311.

[FSMN91] K. Fujikawa, S. Shimojo, T. Matsuura, S. Nishio and H. Miyahara (1991). Multimedia Presentation System 'Harmony' with Temporal and Active Media. In: Proc.s of the Summer 1991 *USENIX* Conference, June, Nashville, TN, 75 - 93.

[GaSM86] L. Nancy Garrett, Karen E., Smith and Norman Meyrowitz (1986). Intermedia: Issues, Strategies, and Tactics in the Design of a Hypermedia Document System. In Proceedings: CSCW '86 (Computer Supported Cooperative Work), 163 - 174.

[GaMP94] F. Garzotto, L. Mainetti and P. Paolini (1994). Adding Multimedia Collections to the Dexter Model. In proceedings: *ECHT '94* (ACM European Conference on Hypermedia Technology), Sep 18-23, Edinburgh UK, 70 - 80.

[Grøn94] Kaj Grønbæk (1994). Composites in a Dexter-Based Hypermedia Framework. In Proceedings: *ECHT '94* (ACM European Conference on Hypermedia Technology), Sep 18-23, Edinburgh UK, 59 - 69.

[GrTr94a] Kaj Grønbæk and Randall H. Trigg (1994). Design issues for a Dexter-based hypermedia system. *Communications of the ACM*, 37 (2), Feb, 40 - 49.

[GrTr94b] Kaj Grønbæk and Randall H. Trigg (1994). Hypermedia system design applying the Dexter model. *Communications of the ACM*, 37 (2), Feb, 26 - 29.

[HKRC92] Bernard J. Haan, Paul Kahn, Victor A. Riley, James H. Coombs and Norman K. Meyrowitz (1992). IRIS Hypermedia Services. *Communications of the ACM*, 35(1), Jan, 36 - 51.

[HaKe97] M. Hakkinen and G. Kerscher (1997). Digital Talking Book Requirements for MML. `http://www.prodworks.com/books/mml-req.html`

[Hala88] Frank G. Halasz (1988). Refl ections on Notecards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31(7), Jul, 836 - 852.

[HaSc94] Frank Halasz and Mayer Schwartz (1994). The Dexter Hypertext Reference Model. *Communications of the ACM*, 37 (2), Feb, 30 - 39. Also NIST Hypertext Standardization Workshop, Gaithersburg, MD, January 16-18 1990.

[HaDH96] W. Hall, H. Davis and G. Hutchings (1996). Rethinking Hypermedia: The Microcosm Approach. Kluwer Academic Publishers, Dordrecht, The Netherlands. ISBN 0-7923-9679-0.

[HaRe94] R. Hamakawa and J. Rekimoto (1994). Object composition and playback models for handling multimedia data. *Multimedia Systems* 2: 26 - 35.

[HTOH96] Komei Harada, Eiichiro Tanaka, Ryuichi Ogawa and Yoshinori Hara (1996). *Anecdote:* A Multimedia Storyboarding System with Seamless Authoring Support. In proceedings: *Multimedia '96* (The Fourth ACM International Multimedia Conference), Boston, MA, Nov, 18 - 22.

[HaWB98] L. Hardman, M. Worring and D.C.A. Bulterman (1998). Integrating the Amsterdam Hypermedia Model with the Standard Reference Model for Intelligent Presentation Systems. To appear in *Computer Standards and Interfaces*. `http://www.cwi.nl/ftp/mmpapers/AHM-SRM.ps.gz`

Bibliography

[HaBu97] L. Hardman and D.C.A. Bulterman (1997). Document Model Issues for Hypermedia. In Handbook of Multimedia Information Management, edited by W.I. Grosky, R. Jain, and R. Mehrotra. Prentice Hall.

[HaBu95a] L. Hardman and D.C.A. Bulterman (1995). Towards the Generation of Hypermedia Structure. In proceedings: First International Workshop on Intelligence and Multimodality in Multimedia Interfaces, July, Edinburgh, UK.

[HaBu95b] L. Hardman and D.C.A. Bulterman (1995). Authoring Support for Durable Interactive Multimedia Presentations. Eurographics '95 State of The Art Report, Aug 1995. `http://www.cwi.nl/ftp/mmpapers/eg95.ps.gz`

[HaRB95] L. Hardman, G. van Rossum and A. van Bolhuis (1995). An Interactive Multimedia Management Game. *Intelligent Systems* 5(2-4), 139 - 150.

[HaBR94] L. Hardman, D.C.A. Bulterman and G. van Rossum (1994). The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. *Communications of the ACM*, 37 (2), Feb, 50 - 62.

[HaBR93a] L. Hardman, D.C.A. Bulterman and G. van Rossum (1993). The Amsterdam Hypermedia Model: Extending Hypertext to Support *Real* Multimedia. Hypermedia 5(1), July, 47 - 69.

[HaBR93b] L. Hardman, D.C.A. Bulterman, and G. van Rossum (1993). Links in Hypermedia: The Requirement for Context. In Proceedings: *ACM Hypertext '93*, Seattle WA, Nov, 183 - 191.

[HaRB93] L. Hardman, G. van Rossum, and D.C.A. Bulterman (1993). Structured Multimedia Authoring. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug, 283 - 289.

[HeKo95] J.L. Herlocker and J.A. Konstan (1995). Commands as Media: Design and Implementation of a Command Stream. In Proceedings: *Multimedia '95*, San Francisco, CA, Nov, 155 - 165.

[HjMi94] R. Hjelsvold and R. Midtstraum (1994). Modelling and Querying Video Data. In Proceedings of the 20th VLDB, Santiago, Chile.

[HoSA89] M.E. Hodges, R.M. Sasnett and M.S. Ackerman (1989). A construction set for multimedia applications. *IEEE Software*, 6(1), Jan, 37 - 43.

[Hosc97] P. Hoschka (ed.) (1997). Synchronized Multimedia Integration Language. W3C Working Draft 09-November-97. Authors: S. Bugaj, D. Bulterman, L. Hardman, J. Jansen, R. Lanphier, N. Layaida, J. Marsh, A. Rao, W. ten Kate, J. van Ossenbruggen, M. Vernick, and Jin Yu.
`http://www.w3.org/TR/WD-smil`

[Hosc97a] P. Hoschka (1997). Synchronized Multimedia.
`http://www.w3.org/AudioVideo/Activity.html`

[HuNi94] S.E. Hudson and C.-N. Hsi (1994). The Walk-Through Approach to Authoring Multimedia Documents. In Proceedings: *Multimedia '94*, San Francisco, CA, Oct, 173 - 180.

[ISO97a] MHEG Part 5. ISO/IEC IS 13522-5. 1997.

[ISO97b] HyTime. Hypermedia/Time-based structuring language. ISO/IEC 10744:1997.

[ISO96] International Standards Organization (1996). Document Style Semantics and Specification Language (DSSSL). ISO/IEC IS 10179:1996.

[JoRo95] R. Joseph, J. Rosengren (1995). MHEG-5: An Overview. `http://www.fokus.gmd.de/ovma/mug/archives/documents/mheg-reader/rd1206.html`

[KMRT96] Tim Krauskopf, Jim Miller, Paul Resnick and Win Treese (1996). PICS Label Distribution Label Syntax and Communication Protocols. Version 1.1. W3C Recommendation 31-October-96. REC-PICS-labels-961031. `http://www.w3.org/PICS/labels.html`

[LeSc94] John J. Leggett and John L. Schnase (1994). Viewing Dexter with Open Eyes. *Communications of the ACM*, 37 (2), Feb, 77 - 86.

[LiBo96] H.W. Lie and B. Bos (1996). Cascading Style Sheets, level 1. W3C Recommendation, Dec. `http://www.w3.org/pub/WWW/TR/REC-CSS1`.

[MaDa89] W. E Mackay and G. Davenport (1989). Virtual video editing in interactive multimedia applications. *Communications of the ACM*, 32(7), July, 802 - 810.

[Macr97] Macromedia. Authorware version 4. Director version 6. `http://www.macromedia.com/`

[NaNa93] J. Nanard and M. Nanard (1993). Should Anchors Be Typed Too? An Experiment with MacWeb. In Proceedings: *ACM Hypertext '93*, Seattle WA, Nov, 51 - 62.

[Niel95] J. Nielsen (1995). Multmedia and Hypertext. AP Professional.

[OTTH92] R. Ogawa, E. Tanaka, D. Taguchi and K. Harada (1992). Design Strategies for Scenario-based Hypermedia: Description of its Structure, Dynamics, and Style. In Proceedings: *ECHT '92* (Fourth ACM Conference on Hypertext), Nov, Milano, Italy, 71 - 80.

[OgHK90] R. Ogawa, H. Harada and A. Kaneko (1990). Scenario-based hypermedia: A model and a system. In proceedings: *ECHT '90* (First European Conference on Hypertext), Nov, INRIA France, 38 - 51.

[OHRE97] J. van Ossenbruggen, L. Hardman, L. Rutledge and A. Eliëns (1997). Style Sheet Support for Hypermedia Documents. In proceedings: *Hypertext '97* (Eighth ACM Conference on Hypertext), Apr 6-11, Southampton UK, 216 - 217.

[OsEl97] J. van Ossenbruggen, L. Hardman and A. Eliëns (1997). A Formalization of the Amsterdam Hypermedia Model. Appendix 1 of this thesis.

[OWL90] OWL International, Inc. (1990). GUIDE™ Hypermedia Information System Version 3.0, User Manual. Current information from InfoAccess: `http://www.infoaccess.com/`

[PaYS90] Murugappan Palaniappan, Nicole Yankelovich and Mark Sawtelle (1990). "Linking Active Anchors: a Stage in the Evolution of Hypermedia," *Hypermedia* 2(1) 47 - 66.

Bibliography

[Pere96] Fernando Pereira (1996). MPEG4: A New Challenge for the Representation of Audio-Visual Information. Keynote speech at Picture Coding Symposium 96. `http://amalia.ist.utl.pt/~fp/artigo54.htm`

[Rada95] R. Rada (1995). Hypertext, multimedia and hypermedia. *The New Review of Hypermedia and Multimedia: Applications and Research* 1, 1 - 21.

[Ragg97] D. Raggett (1997). HTML 3.2 Reference Specification, W3C Recommendation 14-Jan-1997. `http://www.w3.org/TR/REC-html32`

[RTP95] Realtime Transport Protocol (1995). `http://www.cs.columbia.edu/~hgs/rtp/`

[RTSP96] Real-Time Streaming Protocol (1996). `http://www.cs.columbia.edu/~hgs/rtsp/`

[RSVP96] ReSerVation Protocol (1996). `http://www.isi.edu/div7/rsvp/rsvp.html`

[RJMB93] G. van Rossum, J. Jansen, K S. Mullender and D.C.A. Bulterman (1993). CMIFed: a presentation environment for portable hypermedia documents. In Proceedings: *ACM Multimedia '93*, Anaheim CA, Aug, 183 - 188.

[RuDa89] B. Rubin and G. Davenport (1989). Structured content modeling for cinematic information. *SIGCHI Bulletin*, Oct, 21(2), 78 - 79.

[ROHB97a] L. Rutledge, J. van Ossenbruggen, L. Hardman, and D.C.A. Bulterman (1997). Cooperative Use of MHEG-5 and HyTime. Proceedings of *Hypertext and Hypermedia: Products, Tools and Methods*. Paris, France, Sep , 57 - 73.

[ROHB97b] L. Rutledge, J. van Ossenbruggen, L. Hardman, and D.C.A. Bulterman (1997). A Framework for Generating Adaptable Hypermedia Documents. In Proceedings: *ACM Multimedia '97*, Seattle WA, Nov, 121 - 130.

[SFHS91] A. Siochi, E.A. Fox, D. Hix, E.E. Schwartz, A. Narasimhan, and W. Wake (1991). The Integrator: A Prototype for Flexible Development of Interactive Digital Multimedia Applications. *Interactive Multimedia* 2(3), 5 - 26.

[SmZh94] S.W. Smoliar and & H. Zhang (1994). Content-Based Video Indexing and Retrieval. IEEE *Multimedia*, Summer, 62 - 72.

[W3C97] About The World Wide Web Consortium (1997). `http://w3c.org/Consortium/`

[WeWi96] L. Weitzman and K. Wittenburg (1996). Grammar-Based Articulation for Multimedia Document Design. *Multimedia Systems*, 4: 99 - 111.

[West93] N. West (1993). Multimedia Masters: A guide to the pros and cons of seven powerful authoring programs. *MacWorld*, Mar, 114 - 117.

[WBHT97] M. Worring, C. van den Berg, L. Hardman and A. Tam (1997). System Design for Structured Hypermedia Generation. *Lecture Notes in Computer Science* 1306, Visual Information Systems, ed. C. Leung. `http://www.cwi.nl/ftp/mmpapers/LNCS-visual.ps.gz`

[W3C97] World Wide Web Consortium (1997). HTML 4.0 Specification, W3C Working Draft 17-Sep-1997. `http://www.w3.org/TR/WD-html40/cover.html`

A formalization of the Amsterdam Hypermedia Model*

Jacco van Ossenbruggen, Lynda Hardman, Anton Eliëns

email: jrvosse@cs.vu.nl, eliens@cs.vu.nl
Vrije Universiteit, Fac. of Mathematics and Computer Sciences
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

email: Lynda.Hardman@cwi.nl
Multimedia and Human Computer Interaction, CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

A1.1 Introduction

In order to be able to compare the way documents are handled in various hypermedia systems and to be able to define interchange and interoperability standards we need to specify the behavior of such systems and their underlying document model in a non-ambiguous manner.

For hypertext systems, we already have such a non-ambiguous description in the form of the Dexter hypertext reference model [8] and its formal specification in the Z specification language [7]. The Dexter model describes atomic, link, anchor and composition structures in hypertext documents. This provided the hypertext community with a way of comparing the documents created by already existing hypertext systems and to design new systems which followed the (encompassing) Dexter model more closely.

As the use of dynamic media, such as audio and video, in documents increases, and as linking parts of these documents becomes more common, there is a need to be able to compare these more complex hypermedia structures. The Amsterdam hypermedia model (AHM) goes towards describing a hypermedia document including temporal and spatial relationships among constituent media

*This appendix is a preliminary result of work which will be an integral part of the PhD thesis of Jacco van Ossenbruggen, expected to appear in 1998 at the Vrije Universiteit, Amsterdam.

elements, and also pays attention to defining the behavior of links among groups of (dynamic) media items. While Dexter does not specify the internal structure of the information which describes how a component is to be presented at run time, the AHM explicitly defines that part of the presentation information which describes the spatio-temporal layout.

The description of the AHM has, until now, relied on informal descriptions and its (partial) implementation in the CMIFed system. For the model to be more useful to the hypertext community it needs to be described in a precise way. A more formal description of the AHM should identify the exact boundaries of the model, and facilitate the comparison the AHM with other hypermedia models.

This appendix formalizes the AHM model as described in chapter 3 of this thesis. The Object-Z specification language [3, 4, 5] is used in order to be able to present the formal part of the AHM in an incremental and modular way. Object-Z is an object-oriented extension of Z, the language originally used in [7] to formalize the Dexter model.

The formal specification has been developed in close cooperation with the author of this thesis. The specification process helped to abstract from the implementation details of CWI's authoring and play-out environment CMIFed, and to keep the model as generic as possible. Furthermore, the specification process helped to find inconsistencies and design flaws in earlier versions of the model. Parts of the model are not implemented in the CMIFed system, and a formal approach proved to be useful to check especially these parts of the model.

However, the formal specification has not been developed to prove correctness of the model, nor to prove the correct behavior of systems implementing the model. Instead, the specification gives a precise description of the model, which facilitates a deeper understanding of the more complex concepts of the AHM, and allows comparison with other hypermedia models.

The specification given is based upon the description of the AHM as described in this thesis and differs significantly from the AHM as described in [9].

A1.2 Background

The Amsterdam Hypermedia Model provides an abstraction of, and an extension to, the hypermedia model implemented by the CMIFed hypermedia authoring environment. In contrast to many of the hypertext systems which formed the basis of the Dexter model, CMIFed is originally a multimedia system, extended by hyperlink support at a late stage in its development. This firm background in multimedia explains the central role of temporal relationships in the composition mechanisms of the model, the way the behavior of hyperlinks is related to these structures and the explicit modeling of spatio-temporal layout.

When compared to the Dexter model, the main extensions of the AHM are a specific semantics for composite components, its notion of link context and ex-

plicitly defined spatio-temporal layout. The formal specification given in the following sections will focus on these three topics.

Composition structures In contrast to the pure abstract composition facilities of the Dexter model, the AHM defines two specific composition mechanisms: temporal and atemporal composition.

Temporal composition allows the grouping of media items by placing them on the same time axis. This type of composition is common in multimedia systems. Examples include parallel and sequential compositions. By using temporal composition exclusively, the resulting hypermedia document represents a strictly linear multimedia presentation. Hyperlinks can only be used to jump back and forward within the same document, or to start the presentation of another document. Temporal composition does not allow the author to create a hyperlink to parts of the presentation which are optional, and which would not be played if the user never selected the corresponding link.

To allow the inclusion of optional material, the AHM introduces the notion of atemporal composition. Atemporal composition allows grouping of elements which represent alternatives which are accessible by means of hyperlinking. Media objects grouped within an atemporal composite can be part of the main linear stream of the presentation if their initial activation state is $PLAY$ or $PAUSE$. This state can also have the value $INACTIVE$. Objects that are initially paused or inactive require explicit user interaction to be played.

For example, imagine a hypermedia presentation which includes two text entries of a glossary. There are two reasons for not including the entries within a temporal composite, but to use an atemporal composite, combined with an $INACTIVE$ initial activation state. Typically, the entries should only be presented after an explicit request from the user, so both entries are only accessible by link traversal. If they were included in the document by temporal composition only, they would always be presented. Another reason for not using a temporal composite is the absence of obvious temporal relations. While there is a clear structural relation between the two entry components and the glossary component, there is neither a temporal relation between the glossary and the entry components nor a mutual temporal relation between the two entry components. Note that the entries may be displayed in a pop-up window, and in this case there is no spatial relationship between the components. The current version of AHM, however, does not discriminate between spatial and aspatial composition. Temporal and atemporal composition within the AHM will be defined in section A1.6.

Link context In a hypermedia document an end-user can navigate through information by following links defined within the document structure. In an environment with dynamic media, it becomes more important to define the scope of the document affected by the link. To be able to define the scope of a hyperlink

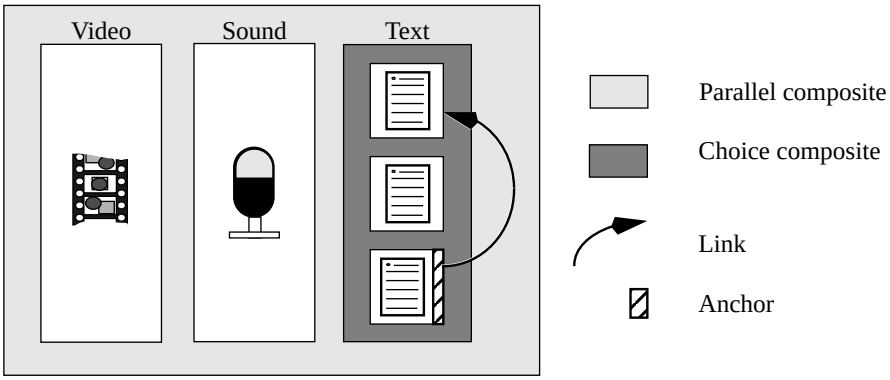


Figure 1: The link between the text atoms may affect the sound and video atoms.

in a declarative way, the AHM introduces the notion of link context [10]. Link contexts make explicit which part of the document stops playing when a link is followed from an anchor and how much of the destination is presented.

For instance, in figure 1, the effect on the sound and video presentation of traversing a hyperlink between two text atoms depends on the link contexts. If the source context is the third text item, and the destination is the first one, the sound and video presentation will continue with no interruption. However, if the source and destination context are defined to be the parallel composite, the presentation of both the sound and video node will be restarted on link traversal. The notion of link context is formalized in section A1.5.

Layout specification The Dexter model provides an adequate way to model structural relationships among the components of a hypermedia document by means of the composite component.

In the Dexter model, all spatial and temporal relationships among components are assumed to be hidden in the presentation specification, arising from the set *PresentSpec*. The presentation specification is the main interface between the storage and the runtime layer. While the internal structure of the *PresentSpec* is considered to be beyond the scope of the model, Dexter makes heavy use of this concept. Each component in the storage layer has a *PresentSpec* to store presentation information local to the component. Additionally, each link-end specifier uses a *PresentSpec*, for instance to store information on how the target should be displayed in the case a link is followed. Note that the link as a whole — being a component itself — also has a *PresentSpec*. To make the situation even more complex, the runtime layer may add an extra presentation specification in order to be able to reflect runtime knowledge in the specification of a component.

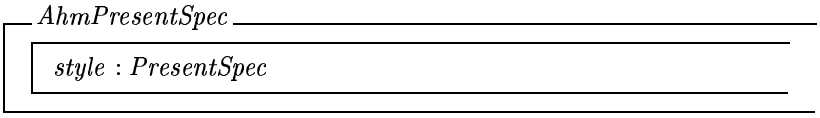
Yet, even this large number of presentation specifications proves to be insufficient in some cases [9]. More importantly, however, we think that the ability to express temporal and spatial relationships between components is too important to be omitted from a hypermedia reference model. As the use of dynamic media will increase, comparing the techniques used to express these relationships becomes an essential part of comparing different hypermedia systems. Additionally, it is useful to have commonly accepted abstraction mechanisms and terminology addressing exactly these topics. For example, one of the main objectives for developing the HyTime standard was the need within the SGML-community for standardized methods of (spatio-temporal) alignment. As such, we see common abstractions for spatio-temporal alignment as a requirement for interoperable hypermedia systems.

A1.3 Preliminaries

Before we start with the specification of the Amsterdam Hypermedia Model, we define the Dexter concepts we reuse in the AHM[1]. Additionally, we need to introduce some basic classes which reflect some (relatively small) differences between the AHM and the original Z specification of the Dexter model.

Presentation Specifications We explicitly discriminate between the information describing temporal and spatial relationships from other presentation information (modeled by $PresentSpec$ as in Dexter). Following Dexter, we consider the inner structure of a $PresentSpec$ beyond the scope of our model. In the following sections, spatio-temporal presentation information is modeled by subclassing $AhmPresentSpec$. All objects which need such a layout specification, need to store other presentation information as well. As a consequence, we include a plain Dexter $presentSpec$ in our $AhmPresentSpec$ to be able to store style information.

$[PresentSpec]$



Anchoring The AHM extends the Dexter anchor by adding a style specification and semantic attributes. In Dexter, anchor style can be modeled in the link-end $Specifier$, but by locating the style information in the anchor, multiple link-ends

[1]We could have used the object-oriented facilities of Object-Z to reuse these parts of the Dexter specification. To make this appendix a self-contained document, we explicitly copied the required definitions of the Dexter specification into the AHM specification.

can share the same anchor style for the same anchor, while link-ends may still override the style specification provided by the anchor. Adding semantic attributes to anchors allows knowledge-oriented applications to store meta data associated with the anchors, and make anchors subject to querying and other information retrieval processing. We leave the Dexter notions of $AnchorValue$ and $Attribute/Value$ pairs unchanged.

$[Attribute, Value]$

$[AnchorValue]$

AhmAnchor

$$anchorStyle : PresentSpec$$
$$attributes : Attribute \pfun Value$$
$$anchorValue : AnchorValue$$

Components We introduce the $AhmComponent$ as a base class for the AHM atom, link and composite components described in section A1.6. All three component classes differ from their Dexter counterparts in having anchors of the $Anchor$ type described above.

AhmComponent

$$attributes : Attribute \pfun Value$$
$$anchors : \mathrm{seq}\ AhmAnchor$$

Specifications Dexter uses the $ComponentSpec$ to indirectly specify a component. The AHM also applies the advantages of this indirect addressing mechanism to media items, anchors and channels. In this way, one can refer to these objects by means of a database query as is already possible for components in the Dexter model. The specifications arise from the following sets:

$[ComponentSpec, MediaItemSpec, AnchorSpec, ChannelSpec]$

Resolver functions are needed to map the specifications to the specified objects. The channel resolver will be introduced in section A1.4.1. We follow the Dexter convention of representing the raw media data by the given set $Atomic$.

$[Atomic]$

$$
\begin{array}{|l}
mediaResolver : MediaItemSpec \pfun Atomic \\
compResolver : ComponentSpec \pfun \downarrow AhmComponent \\
anchorResolver : AnchorSpec \pfun \downarrow AhmAnchor
\end{array}
$$

Note that the resolvers no longer return identifiers (such as Dexter's Uid and $AnchorId$). We consider the use of explicit identifiers and accessor functions superfluous since we can use the implicit object identifiers of the Object-Z language.

A1.4 Spatio-Temporal Layout

Spatial layout is modeled differently from temporal layout in the AHM. Spatial layout definitions are defined by $Channel$s. Components can share the same spatial layout by using the same $Channel$. Temporal layout definitions are modeled by synchronization arcs ($SyncArcs$). As noted before, temporal relations also play an important role in document composition.

A1.4.1 Spatial relationships

In the AHM, spatial relationships between components are defined by the use of channels, which are abstract output devices for playing the contents of components. When the document is played, channels are mapped onto physical output devices, so they can be effectively used for resource allocation purposes. Channels may additionally define other (default) presentation attributes for all associated components. For example, an author is able to change the font of all captions in the document, by changing the font of the "caption-channel", instead of changing the presentation specification of all individual caption components.

In the model, spatial constraints are supported by the $Channel$ attribute of the components. Channels are defined by a presentation specification and a resource allocation information field.

$$[ResourceSpec]$$

$Channel$

$$
\begin{array}{|l}
parent : ChannelSpec \\
style : PresentSpec \\
resourceSpec : ResourceSpec \\
attribute : Attribute \pfun Value
\end{array}
$$

$$
\begin{array}{|l}
channelResolver : ChannelSpec \pfun Channel
\end{array}
$$

A1.4.2 Temporal relationships

Temporal relationships among the components in the AHM are represented by synchronization arcs. Temporal constraints indicate a preferred delay and allowable deviations. The constraints can be $ADVISORY$, meaning that realization of the constraint at runtime is desirable, but not strictly necessary, or $HARD$, meaning that violating the constraint would be an error. The initial values default to an (advisory) delay of zero.

$$SyncType ::= HARD \mid ADVISORY$$

TemporalConstraint

$preferredTime : \mathbb{R}$
$minimimBeforeTime : \mathbb{R}$
$maximumAfterTime : \mathbb{R}$
$type : SyncType$

INIT

$preferredTime = minimimBeforeTime = maximumAfterTime = 0$
$type = ADVISORY$

A synchronization arc is defined by references to the anchors of the arc's source and destination, followed by the temporal constraint between these components. Synchronization arcs are used to denote temporal constraints among descendants of a composite document, and are considered to be part of the composite's presentation specification (see also section A1.6.3). Constraints may be defined between the two intervals associated with the anchors. Note that any of the thirteen possible temporal relations defined by Allen [2] can be described by at most two synchronization arcs (see also chapter 3 of this thesis). Additionally, constraints may be defined on specific points in the document as well. For instance, a synchronization arc may be used to synchronize a video frame with an audio sample. In that case the anchors resolve to an individual frame or sample.

Note that we explicitly do not overload the Dexter link for specifying temporal constraints because hyperlinks are considered primarily to describe semantic relationships (although they can be used for navigation purposes). In contrast, synchronization arcs cannot be used for describing semantic relationships, nor for navigation, but are used for describing temporal presentation information. Both links and synchronization arcs, however, use the same media-independent anchoring mechanism to address their end points.

SyncArc

$source : (ComponentSpec \times AnchorSpec)$
$destination : (ComponentSpec \times AnchorSpec)$
$constraint : TemporalConstraint$

$compResolver(first(source)) \neq compResolver(first(destination))$

INIT

$constraint.INIT$

A1.5 Link Context

The declarative aspects of the concept of link context can be easily formalized by deriving a new class from the Dexter *Specifier*. We need a flag which indicates whether the source context needs to be continued, paused or deactivated:

$$SourceActivation ::= CONTINUE \mid PAUSE \mid DEACTIVATE$$

The destination context can be activated in a playing or a paused state:

$$DestinationActivation ::= PLAY \mid PAUSE$$

We reuse the Dexter specification for modeling *Direction*:

$$Direction ::= FROM \mid TO \mid BIDIRECT \mid NONE$$

The presentation specifier for the link end specifier contains both flags, and inherits from the *AhmPresentSpec*:

SpecifierPresentSpec

AhmPresentSpec

$srcActivation : SourceActivation$
$dstActivation : DestinationActivation$

INIT

$srcActivation = CONTINUE$
$dstActivation = PLAY$

For each link end (i.e. specifier), the context component of the link is specified by an attribute of type *ComponentSpec*. The context component is typically a composite containing the link end component. For example, if the source of a link is

an anchor in a subtitle component, the associated context is likely to be the composite containing the subtitle along with the video and sound track component.

LinkSpecifier

$$
\begin{array}{l}
presentSpec : SpecifierPresentSpec \\
anchor : ComponentSpec \times AnchorSpec \\
context : ComponentSpec \\
direction : Direction \\
\hline
direction \neq NONE
\end{array}
$$

INIT

$$
\begin{array}{l}
context = first(anchor) \\
presentSpec.INIT \\
direction = FROM
\end{array}
$$

By default, the specifier is initialized to represent the most simple case, i.e. where the context component equals the component containing the anchor.

The context's role (whether it is a source or destination context) depends on the direction of the associated specifier ($FROM$ or TO respectivly). Actually, the context can act as both source and destination context (if $direction = BIDIRECT$) so in general, the contexts of a link can only be determined at run-time. Note that Dexter's use of $NONE$ has been criticized [6] because of its undefined semantics. In AHM, we disallow a $NONE$ direction.

By making the context itself not a part of the presentation specification, we claim that context adds structural (and indeed semantic) information to a hyperlink and is considered to be more than "just" presentation information. Furthermore, by requiring the context of a link-end to be a (composite) component, contexts themselves are closely related to the document structure. To promote anchor reuse, our notion of context is defined on the link level, and not on the anchor level, as is used in MacWeb [11].

A1.6 Components in the AHM

Keeping the above descriptions of spatio-temporal relationships in mind, we can now formalize the various components of the AHM. We describe the AHM atomic, link and composite component, focusing on the structure of the spatio-temporal information within the components. Additionally, we discuss the difference between the Dexter and AHM components and their specifications.

A1.6.1 Atoms

The AHM atomic component mirrors its Dexter counterpart, but makes its spatio-temporal characteristics explicit. We expect all spatio-temporal arithmetic to be carried out using real numbers (the associated unit used, e.g. seconds, milliseconds, pixels or centimeters, is outside the scope of the model). The duration and layout information of the atomic component is described in its presentation specification, since it provides layout information which needs to be interpreted in the runtime layer.

Presentation Specification of Atoms The AHM atom's temporal characteristics are reflected by its duration (stated by the author or as an intrinsic property of the media item itself). Its spatial layout is defined by the associated channel. An atom may modify its channel's layout definition by defining an (smaller) extent within the extent defined by the channel. This is reflected by the extent and position attributes. Style information is modeled by the inherited style attributed.

AtomPresentSpec

AhmPresentSpec

$$
\begin{aligned}
&duration : \mathbb{R} \\
&channel : ChannelSpec \\
&position : \mathbb{R} \times \mathbb{R} \\
&extent : \mathbb{R} \times \mathbb{R}
\end{aligned}
$$

Anchoring in Atoms The concept of anchoring for atomic components needs to be extended to include a duration. For continuous media items, the media dependent $AnchorValue$ can be expected to define the duration of an anchor (for example, by defining a range of frames for a video fragment). But for static media such as text, we need to define the interval in which the anchor is active. Additionally, we have extended the Dexter anchor with attributes to store semantic information (these can be used for information retrieval or knowledge representation purposes) and an anchor style presentation specification. These additions are inherited from the $AhnAnchor$ base class:

AtomicAnchor

AhmAnchor

$$
\begin{aligned}
&startTime : \mathbb{R} \\
&duration : \mathbb{R}
\end{aligned}
$$

The Atomic Component The AHM atomic component contains the presentation specification described above, and a list of anchors. The actual media content is referred to by a system dependent media item reference (this can be a filename, URL or database query) and a media dependent anchor value denoting which part of the media item is used. In this way, the model is independent of the granularity of the server providing the media items (e.g. an author does not need to create a new image file if only a part of that image needs to be included in the presentation).

$$
\begin{array}{|l}
\hline
AhmAtom \\
\upharpoonright (Init, attributes) \\
AhmComponent \\
\quad \begin{array}{|l}
\hline
presentSpec : AtomPresentSpec \\
anchors : \text{seq } AtomicAnchor \\
content : MediaItemSpec \times AnchorValue \\
\hline
\end{array} \\
\hline
\end{array}
$$

A1.6.2 Links

The link is — following the Dexter model — a component with a sequence of link-end specifiers derived from *Specifier*. It can be used to define links of arbitrary arity. Because the AHM does not associate spatio-temporal information with the link component itself, there is no need to specify a new presentation specification class for link components. Other style information, for instance to display the link in a link browser, is specified using a Dexter *PresentSpec*. Being a component, links can be end points of other links, so a link needs anchors just as the other components do. The link component inherits its attributes and anchors from AhmComponent. Its class schema adds a style (e.g. to display the link in a link browser), a duration (of a possible transition from source to destination), a position (the position of the destination relative to the source anchor or mouse click) and a list of specifiers. Note that the link contains only one duration and position attribute, which is not sufficient for links containing multiple destinations. This problem needs to be solved in a next version of the model.

AhmLink

$\upharpoonright(Init, attributes, anchors)$
AhmComponent

$style : PresentSpec$
$duration : \mathbb{R}$
$position : \mathbb{R} \times \mathbb{R}$
$specifiers : \text{seq } LinkSpecifier$

$\#specifiers \geq 2$

Constraining the number of end point specifiers to be at least two might prove to be too restrictive, especially in an open collaborative work environment [6].

A1.6.3 Composites

The AHM discriminates between two types of composition: temporal and atemporal. All children of a temporal composite share the same time line and need to be synchronized using synchronization arcs. In contrast, the children of an atemporal composite are scheduled on different time lines and cannot have any temporal relationships.

Anchoring in Composites The AHM addresses the underspecification of anchors for composite components [6] by defining the anchor value to be a list of references to anchors defined by the descendants of the composite. See figure 2. This can be used to group anchors by building a hierarchical structure of composite anchors. Semantic attributes and style information can be attached to each anchor.

CompositeAnchor

AhmAnchor

$anchorValue : \text{seq}(ComponentSpec \times AnchorSpec)$

Presentation Specification of Composites Because of their different temporal properties, the two composite types need different presentation specifications. The temporal presentation specification contains a list of synchronization arcs defining the temporal relations among the children of the composite. By specifying a duration a user can override the intrinsic duration of the composite (the playing environment may scale or clip the children in order to achieve this):

Appendix 1

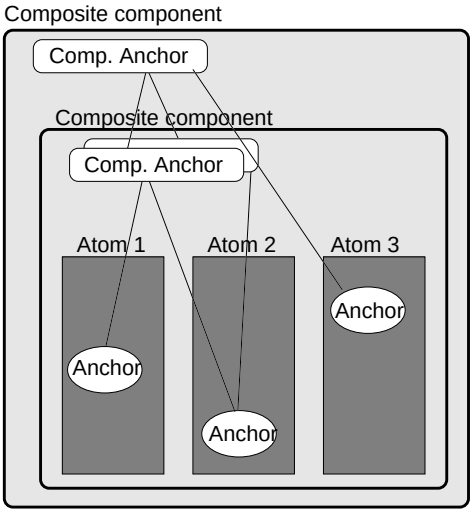


Figure 2: Composite anchoring hierarchy

TemporalPresentSpec
AhmPresentSpec

$duration : \mathbb{R}$
$syncArcs : \text{seq } SyncArc$

Since there are no temporal relationships between the children of an atemporal composite, the atemporal presentation specification does not contain any synchronization arcs. Instead, it specifies the initial activation state of each of the children. This state can be play, pause or inactive. Inactive children can only made active as a result of link traversal:

$$ActivationState ::= PLAY \mid PAUSE \mid INACTIVE$$

AtemporalPresentSpec
AhmPresentSpec

$initialStateOfChildren : \text{seq } ActivationState$

The Composite Component The *AhmComposite* serves as an abstract base class for the two composites:

$$
\begin{array}{|l}
\hline
\textit{AhmComposite} \\
\upharpoonright(\textit{Init}, \textit{attributes}) \\
\textit{AhmComponent} \\
\hline
\textit{anchors} : \text{seq } \textit{CompositeAnchor} \\
\textit{children} : \text{seq } \textit{ComponentSpec} \\
\Delta \\
\textit{descendants} : \mathbb{P} \downarrow \textit{AhmComponent} \\
\hline
\forall\, a : \text{ran } \textit{anchors} \bullet \forall\, \textit{avalue} : \text{ran } a.\textit{anchorValue} \bullet \exists\, d : \textit{descendants} \bullet \\
\quad \textit{compResolver}(\textit{first}(\textit{avalue})) = d \land \\
\quad \textit{anchorResolver}(\textit{second}(\textit{avalue})) \in \text{ran } d.\textit{anchors} \\
\hline
\end{array}
$$

Note that the state invariant requires that the anchors of the composite refer to existing anchors of the composite's descendants (the formal definition of *descendants* is left out for reasons of brevity) The temporal composite extends the *AhmComposite* with this presentation specification:

$$
\begin{array}{|l}
\hline
\textit{TemporalComposite} \\
\textit{AhmComposite} \\
\hline
\textit{presentSpec} : \textit{TemporalPresentSpec} \\
\hline
\end{array}
$$

The specification of the atemporal composite mirrors the definition of its temporal counterpart. It requires the specification of an initial state for all its children:

$$
\begin{array}{|l}
\hline
\textit{AtemporalComposite} \\
\textit{AhmComposite} \\
\hline
\textit{presentSpec} : \textit{AtemporalPresentSpec} \\
\hline
\#\textit{children} = \#\textit{presentSpec.initialStateOfChildren} \\
\hline
\end{array}
$$

A1.7 Hypermedia

Finally we define the *Hypermedia* class. The composition structure in AHM, as in Dexter, specifies a directed, acyclic graph. The AHM differs, however, by requiring that the graph should have a unique root element. If a graph has multiple potential root elements, these can always be combined together into a single root element using atemporal composition.

The first state invariant follows the constraints of the Dexter hypertext formalization in order to ensure accessibility of the components. It states that every component should be accessible by the external component resolver (ran $compResolver = components$). Furthermore, every component needs to be a descendant of the unique $root$ component ($components = root.descendants$).

The Dexter hypertext specification contains several consistency constraints which can only be ensured in a "closed" hypermedia system. In particular, Dexter requires all links to refer to existing components and anchors, *within* the system. As a result, deleting a component involves deleting all links resolving to the deleted component. Since these constraints can never be ensured in a open environment (such as the WWW), they have been left out in the following specification.

Hypermedia

$root : AhmComposite$
Δ
$components : \mathbb{F} \downarrow AhmComponent$

$components = \varnothing \lor components = root.descendants$
ran $compResolver = components$

INIT

$components = \varnothing$

A1.8 Conclusion

The abstractions concerning the mechanisms used to describe spatio-temporal relationships and the context of a hyperlink are important features of a hypermedia reference model. The Amsterdam hypermedia model provides extensions to the Dexter model that address these issues. Previous descriptions of the model gave only informal descriptions of these abstractions, and so we have expressed them here as part of a formal description of the model using the specification language Object-Z. In the process of formalizing the model, we have obtained the refined version of that presented in [9], as presented in this thesis, where a number of flaws have been fixed.

In order to formalize the operational behavior of the additions to the storage layer as presented in this appendix, the formalization of the Dexter runtime layer needs to be extended The specification of the Dexter runtime layer, as given in in [7] focuses on the mechanics of link traversal. A specification of the AHM runtime layer should include the effect of context upon link traversal, and model the temporal interdependencies between the possible state transitions within the model.

References

[1] ACM. *Proceedings of ACM Hypertext '93 (Seattle)*, November 1993.

[2] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–844, November 1983.

[3] Roger Duke, Paul King, Gordon Rose, and Graeme Smith. The Object-Z Specification Language: Version 1. Technical Report 91-1, Software Verification Research Centre, Department of Computer Science, The University of Queensland, Australia, April 1991. The most complete (and currently the standard) reference on Object-Z. It has been reprinted by ISO JTC1 WG7 as document number 372.

[4] Roger Duke and Gordon Rose. Modelling Object Identity. Technical Report 92-11, Software Verification Research Centre, Department of Computer Science, The University of Queensland, Australia, 1992.

[5] Roger Duke, Gordon Rose, and Graeme Smith. Object-Z: a Specification Language Advocated for the Description of Standards. Technical Report 94-45, Software Verification Research Centre, Department of Computer Science, The University of Queensland, Australia, December 1994.

[6] K. Grønbæck and R. Trigg. Design Issues for a Dexter-Based Hypermedia System. *Communications of the ACM*, 37(2):40–49, February 1994.

[7] F. Halasz and M. Schwarz. The Dexter Hypertext Reference Model. In *NIST Hypertext Standardization Workshop*, pages 95–133, January 1990.

[8] F. Halasz and M. Schwarz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, February 1994. Edited by K. Grønbæck and R. Trigg.

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

[10] L. Hardman, D.C.A. Bulterman, and G. van Rossum. Links in hypermedia: the requirement for context. In *Proceedings of ACM Hypertext '93 (Seattle)* [1], pages 183–191.

[11] Jocelyne Nanard and Marc Nanard. Should Anchors Be Typed Too — An Experiment with MacWeb. In *Proceedings of ACM Hypertext '93 (Seattle)* [1], pages 51–62.

when the cursor is over the anchor marker, or when the mouse button is depressed.
Anchors each have an identifier and a value, but no individual presentation specifications, and in particular no duration for an anchor. Video and audio anchors are not supported.

Content is normally specified via a file, but text can be included directly in the document.

An atomic component is required to have associated content and a channel in order to be played.

A2.3 Composite components

The composite types in CMIFed are parallel, serial and choice. Parallel and serial are forms of temporal composition; choice is a form of atemporal composition. There is no presentation specification or semantic attributes applicable to a composite as a whole. Anchors and children can be specified.

A composite can have anchors, but these function only as the destination of a link. The anchors refer implicitly to the beginning of the complete composite. Composite anchors are not supported.

Children are contained within a composite and cannot be referred to from other composites. Children are atomic or composite components; they cannot be link components.

Synchronization arcs can be specified between any two atomic components that belong to the same multimedia presentation. The synchronization arc specifies the source and destination components, a BEGIN or END specifier, and a time delay. Anchor references and other scheduling information are not supported.

A2.4 Link component

Of all the parts that can be specified in a link in the model, a link in CMIFed has only source and destination component reference/anchor reference pairs and a direction. In other words, the link component has no presentation specification, no attributes, no anchors and only two specifiers.

The link source is an anchor in an atomic component (otherwise the reader cannot interact with it). The destination is an anchor in either an atomic or a composite component. The direction can be FROM, TO or BIDIRECT, the last only if the destination is an anchor within an atomic component. The direction applies to the link, rather than each specifier, which for the case of a single source/single destination link is equivalent.

Reference

[ROHB97b] L. Rutledge, J. van Ossenbruggen, L. Hardman, and D.C.A. Bulterman (1997). A Framework for Generating Adaptable Hypermedia Documents. In Proceedings: *ACM Multimedia '97*, Seattle WA, Nov.

A2 The AHM as Implemented in CMIFed

The hypermedia authoring environment CMIFed was originally based on the CMIF model but has since been updated to include the main elements of the Amsterdam hypermedia model. We state here how the model supported by the editor compares with the full AHM. A complete specification of the model encapsulated in the CMIF document format in terms of HyTime is given in [ROHB97b].

A2.1 Channel

CMIFed implements the channel as defined in the model, except that the semantic attributes are not recorded explicitly. Channels can be of two types: layout channels which contain other channels and media channels. Layout can thus be specified hierarchically. The highest-level channel is a window. Styles which can be assigned are media item and anchor styles. Transition styles are not supported.

A2.2 Atomic component

CMIFed implements the atomic component as defined in the model to a large extent. Presentation specifications, anchors and content are supported, although semantic attributes are not.

The presentation specification consists of a channel reference, duration, scaling and style information. A channel specifying spatial and style information has to be assigned to the component. The position of the content in relation to the channel cannot be specified—images and video are displayed in the centre of the channel, text is started at the top. The duration of the component is derived from the associated content (in the case of video or audio), can be derived from the composition structure surrounding the component (for text and images), and can be specified explicitly by the author (text and images). Content for image items can be scaled to fit the channel or given a scale factor in terms of the image's intrinsic size. Styles which can be assigned are media item style and anchor style and can override defaults associated with the channel. Anchor style does not include changing the style of the anchor marker depending on the interactive behaviour of the reader, for example by changing the anchor style

Samenvatting

Hypermedia presentaties zijn samengesteld uit tekst, videobeelden, plaatjes en geluidsfragmenten, waarbij een lezer kan springen naar verschillende presentaties. Fig. 1 laat een voorbeeld zien van zo'n hypermedia presentatie. Dit voorbeeld toont diverse elementen die gecombineerd zijn in meerdere presentaties, zoals de drie scènes in de figuur. Tevens zijn er keuzepunten van waaruit een lezer 'door kan klikken' naar andere presentaties. Deze zijn weergegeven door de omkaderde woorden in Fig. 1. We gebruiken de term *media-items* voor de stukjes tekst, video, etc. De overgang naar een andere presentatie wordt een *hyperlink* genoemd. Wanneer media-items worden samengevoegd tot een presentatie, wordt met behulp van temporele relaties tussen de items aangegeven wanneer en voor hoe lang de items op het scherm moeten verschijnen.

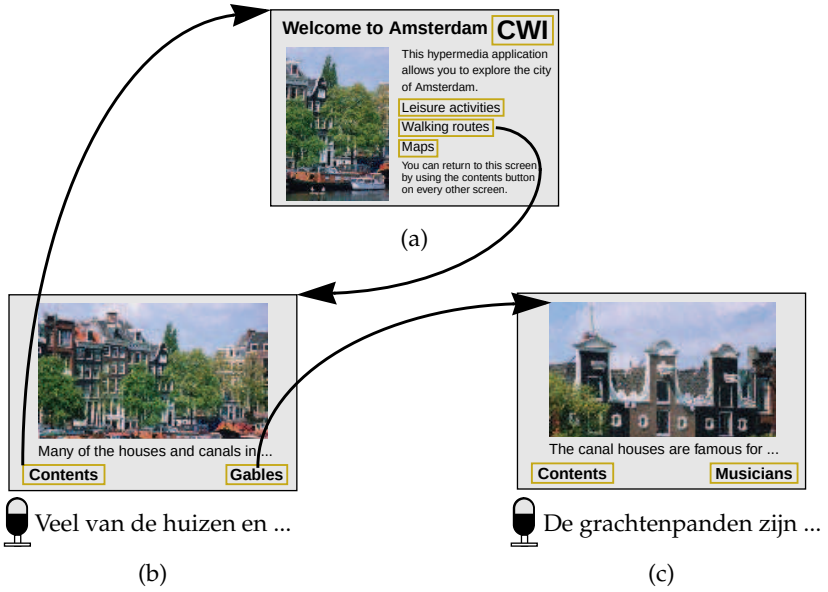


Figure 1. An example hypermedia presentation

Een hypermedia presentatie wordt dynamisch gegenereerd op het moment dat het onderliggende *document* dat de verschillende aspecten van de presentatie specificeert wordt afgespeeld. Bij voorbeeld, de presentatie afgebeeld in Fig. 1(a) is gegenereerd uit een document dat twee stukjes tekst bevat en twee plaatjes die allemaal tegelijkertijd afgespeeld moeten worden.

Een *documentmodel* beschrijft wat de mogelijke onderdelen zijn van het onderliggende document. Een vergelijking is met tekstverwerkers, waar een document kan bestaan uit alinea's, pagina's, titels enz. Het documentmodel van een tekstverwerker is dus de verzameling van deze objecten en hun mogelijke relaties.

Zo'n documentmodel is nuttig om presentaties af te spelen in verschillende omgevingen. Dit kan weer worden vergeleken met tekstverwerkers, waarbij auteurs gebruik kunnen maken van een tekstverwerker naar keuze, terwijl de documenten toch tussen verschillende tekstverwerkers kunnen worden uitgewisseld. Een ander voordeel van een expliciet model is dat de documenten ook kunnen worden gebruikt voor andere doeleinden dan alleen het bekijken ervan, zoals het creëren van meerdere versies voor verschillende uitvoerapparaten, of om de visuele stijl van het document te veranderen.

Op basis van een model van hypermedia documenten kan een auteursysteem worden ontwikkeld dat ondersteuning biedt bij het creëren en wijzigen van de onderdelen van het hypermediadocument.

Dit proefschrift beschrijft de vereisten van een documentmodel voor hypermedia en definieert zo'n model. Vervolgens worden de gebruikers-interfaces van bestaande auteursystemen voor hypermedia documenten geanalyseerd. Op basis van deze analyse en van het model worden dan de vereisten voor een complete auteursomgeving vastgesteld. Tenslotte wordt het CMIFed systeem beschreven, een hypermedia auteursysteem, geïmplementeerd door medewerkers van de CWI multimedia groep.

Vereisten voor een model voor hypermedia

Als we nog eens kijken naar het voorbeeld in Fig. 1, kunnen we op basis daarvan een aantal vereisten beschrijven voor een hypermedia documentmodel.

Ten eerste kunnen we vaststellen dat in een hypermedia presentatie sprake is van meerdere media-items. Het moet daarom mogelijk zijn om aan te geven welke items deel uit moeten maken van een presentatie en welk deel van de items moet worden getoond. Dit laatse is nuttig om duplicatie van vergelijkbare media-items te vermijden. Het is ook nodig dat het data type van elk media item bekend is, zodat het afspeelprogramma weet hoe het media-item moet worden behandeld.

Elk media-item wordt getoond op een bepaalde positie op het beeldscherm en heeft bepaalde afmetingen. De titel in Fig. 1(a) bevindt zich bijvoorbeeld aan de linkerbovenkant van het scherm en overlapt het CWI logo rechts daarvan niet.

Ook tijds-informatie is nodig, bijvoorbeeld om aan te geven dat alle vier de items in Fig. 1 tegelijkertijd getoond moeten worden. Elk media item heeft daarom een geassocieerd startijdstip en een speelduur.

Naast informatie per media-item, moet ook informatie kunnen worden vastgelegd die meerdere items betreft. De scène getoond in Fig. 1(a) is bijvoorbeeld samengesteld uit vier afzonderlijke items. Middels het documentmodel moet ook dit type informatie kunnen worden vastgelegd.

Ook over hyperlinks moet informatie kunnen worden vastgelegd. Er moet bijvoorbeeld kunnen worden aangegeven waar de lezer kan klikken op het scherm (bijvoorbeeld op de omkaderde woorden in Fig. 1) en wat de bestemming is van de hyperlink. Hyperlinks in hypermedia kunnen vrij complex zijn omdat de presentaties bestaan uit meerdere items, waarvan sommige een 'doorlopend' karakter hebben (bijvoorbeeld video en audio). Als een lezer een hyperlink volgt kan het zo zijn dat slechts een deel van de presentatie verandert. Bijvoorbeeld, in Fig. 1 (b) en (c) blijft het *Contents* tekst item onveranderd wanneer de hyperlink vanuit het *Gables* tekst item word gevolgd.

Alhoewel het documentmodel in de eerste plaats bedoeld is om er voor te zorgen dat een presentatie gereproduceerd kan worden op basis van de opgeslagen informatie, kan het ook worden gebruikt om te zoeken naar onderdelen van een document. Elk media-item, of deel ervan, kan een object of concept uit de wereld representeren. Daarom kunnen in het documentmodel media-onafhankelijke beschrijvingen gegeven worden van de informatie in de media-items. De plaatjes in Fig. 1 zouden bijvoorbeeld de beschrijving "stadsgezicht" kunnen hebben.

Dit zijn dus de vereisten voor een documentmodel voor hypermedia. In het proefschrift laten we zien dat bestaande documentmodellen voor hypertekst (in het bijzonder het Dexter model) en voor multimedia (het CMIF model) niet voldoen aan alle vereiste aspecten voor hypermedia. Met name de volgende aspecten ontbreken:

- De mogelijkheid om te specifi ceren welke delen van een presentatie veranderen wanneer een hyperlink wordt gevolgd. Moet dan bijvoorbeeld de gehele presentatie worden veranderd, alleen een deel ervan, of zou de nieuwe presentatie moeten worden afgespeeld tezamen met de bestaande presentatie?
- De mogelijkheid om vast te leggen volgens welke stijl de bron van een hyperlink moet veranderen in de bestemming. Moet dan het scherm meteen in de nieuwe presentatie veranderen, moet er een kleine tussenpauze komen of moet de eerste presentatie geleidelijk overgaan in de nieuwe presentatie?
- De mogelijkheid om media-onafhankelijke beschrijvingen van onderdelen van media-items op te nemen. Zo zou bijvoorbeeld in Fig. 1 het concept 'huis' geassocieerd kunnen worden met verschillende delen van de plaatjes.

In het proefschrift wordt op basis van deze tekortkomingen geconcludeerd dat er behoefte is aan een nieuw model voor hypermedia.

Een hypermedia document model—AHM

In hoofdstuk 3 wordt het Amsterdam Hypermedia Model (AHM) gedefi nieerd. Dit model voldoet aan de in hoofdstuk 2 beschreven vereisten. De basiselementen van het model zijn de atomaire component, de samengestelde component en de link component alsmede het kanaal. Deze basiselementen bestaan weer uit sub-elementen die ervoor zorgen dat het model de vereiste expressiviteit bezit.

Atomaire component
Een atomaire component heeft alle eigenschappen die kunnen worden geassocieerd met individuele media-items, inclusief een referentie naar het media-item zelf. Deze eigenschappen zijn: tijdsduur, spatiële informatie, stijl, en media-onafhankelijke beschrijvingen van de inhoud van het item. Tevens kan worden aangegeven welke delen van het media-item kunnen worden gebruikt als begin- en eindpunten: de zogenaamde *ankers*. Een anker vereist een data-afhankelijke specifi catievan een deel van het media-item, en kan ook eigenschappen hebben.

Samengestelde component
Een samengestelde component maakt het mogelijk om een aantal andere componenten te groeperen in één element dat gebruikt kan worden op dezelfde manier als een atomaire component. Er zijn twee soorten samengestelde componenten: temporele en a-temporele. Bij een temporele samengestelde component kunnen temporele relaties worden aangegeven tussen de verschillende samenstellende delen: de synchronisatie verbindingen. Bij een a-temporele samengestelde component wordt geen gebruik gemaakt van voor-gespecifi - ceerde temporele relaties. Dit maakt het mogelijk om scènes te creëren waarin de lezer zelf kan navigeren. Hiervoor beschikt een a-temporele samengestelde component over activatie-informatie, die aangeeft welke van de samenstellende delen geactiveerd moet worden wanneer de samengestelde component wordt geactiveerd (bijvoorbeeld door het volgen van een hyperlink).

Link component
Een link component specifi ceertde bron- en bestemmingscomponenten van een hyperlink. De link bestaat uit een aantal *specifiers* die kunnen fungeren als bron of bestemming van een hyperlink. Elke specifi erheeft een referentie naar een anker binnen een component. Dit anker kan de plaats zijn waar een lezer klikt, of het kan oplichten wanneer men bij de bestemming van de link aankomt. Verder geeft de specifi eraan welke delen van de presentatie verdwijnen of verschijnen als de link gevolgd wordt.

Kanaal
Het model bevat ook een kanaal-element. Een kanaal voegt spatiële, stijlistische en data informatie samen in een vorm die herbruikbaar is door verschillende atomaire componenten.

Door het documentmodel af te leiden van een voorbeeldpresentatie tonen we in het proefschrift aan dat alle onderdelen van het model noodzakelijk zijn om een hypermedia presentatie te beschrijven. Om te demonstreren dat de onderdelen ook voldoende zijn laten we zien dat het AHM in staat is om de modellen te beschrijven welke impliciet aanwezig zijn in een reeks van hypertekst- en hypermediasystemen. Op basis hiervan wordt geconcludeerd dat het AHM een compleet doch niet te complex model is voor hypermedia documenten.

Paradigma's voor het ontwikkelen van hypermedia-documenten

Er bestaan een aantal verschillende benaderingen om multimedia-documenten te ontwikkelen. In het proefschrift analyseren we een selectie van bestaande auteursystemen en categoriseren hun benaderingen in een aantal paradigma's. Deze paradigma's kunnen worden beschouwd als de uitgangspunten waarop de gebruikersinterfaces zijn gebaseerd.

De paradigma's worden beoordeeld op hun geschiktheid voor vier verschillende onderdelen van de auteurstaak: het creëren van een narratieve structuur, het vastleggen van temporele relaties, het specifi ceren van een spatiële layout en het defi niëren van links tussen de individuele presentaties. Structuurgebaseerde systemen zijn—zoals de naam al suggereert—geschikt voor het editen van de structuur van een presentatie. Indien de structuur ook de temporele relaties weerspiegelt is dit paradigma ook geschikt om temporele aspecten van een presentatie vast te leggen. Tijdsbalk-gebaseerde systemen zijn vooral geschikt om de temporele relaties van een presentatie weer te geven. Systemen gebaseerd op flowcharts en scripts zijn daarentegen goed in het specifi ceren van een meer algemene interactie, waarbij moet worden gezegd dat systemen gebaseerd op flowcharts een betere gebruikersinterface hebben. Geen enkele van deze paradigma's is echter geschikt voor het editen van de layout van een presentatie; noch voor het creëren van links. Hierbij moet echter worden opgemerkt dat het structuurgebaseerde paradigma het mogelijk maakt om verschillende delen van een link te specifi ceren.

We kunnen dus concluderen dat elk van de paradigma's het meest geschikt is voor een specifi eke edit-taak, maar dat geen enkel paradigma voldoende krachtig is om alle edit-aspecten van een hypermedia presentatie te ondersteunen. Een systeem dat de auteurstaak volledig ondersteund vereist dus meerdere gebruikersinterfaces, maar wel binnen een samenhangende omgeving.

Vereisten voor het ontwikkelen van hypermedia presentaties

We specifi ceren in het proefschrift de functionele vereisten van een auteursysteem dat het AHM documentmodel ondersteunt. Om overzicht te houden is de auteursomgeving in vier lagen verdeeld: de data-laag, de resource-laag, de component-laag en de document-laag.

De data-laag bevat de media-items, en schermt daardoor de andere lagen af van afhankelijkheden van het data-formaat.

De resource-laag bevat de resources die gebruikt worden voor verschillende aspecten van het document. Voorbeelden hiervan zijn de data-formaat-resource, welke gebruikt wordt om data te interpreteren, stijl informatie voor lettertypen, media-onafhankelijke beschrijvingen van een toepassingsgebied, en layout informatie. Het nut van het toevoegen van de resource-laag als een afzonderlijke laag is dat elke resource op deze wijze kan worden vervangen door een vergelijkbare resource terwijl de documenstructuur zelf onveranderd blijft. Dit maakt het mogelijk om meerdere presentaties te genereren uit dezelfde onderliggende documentstructuur. Layout kan zo bijvoorbeeld worden aangepast voor bepaalde uitvoerapparatuur.

De component-laag bevat de atomaire, samengestelde en link componenten van het document. Het nut van deze laag is dat de atomaire en samengestelde componenten uniform behandeld kunnen worden in de auteursomgeving.

Temporele en spatiële layout spelen een zeer belangrijke rol in hypermedia presentaties. De document-laag maakt het mogelijk om deze aspecten te specifi ceren en te wijzigen en geeft deze informatie vervolgens door aan de component-laag en de resource-laag. In het proefschrift worden de vereisten voor de document-laag besproken en worden voorbeelden gegeven van mogelijke gebruikersinterfaces voor het editen van temporele en spatiële informatie. Bijzondere aandacht is hierbij gegeven aan tijdsbalk-illustraties, tempowisselingen en het navigeren langs de presentatie-tijdsbalk.

Een hypermedia auteursomgeving—CMIFed

Nadat de vereisten voor een auteursomgeving zijn geanalyseerd, wordt CMIFed beschreven, een systeem geïmplementeerd voor het creëren van hypermedia documenten. Het belang van CMIFed vanuit het perspectief van dit proefschrift is dat dit systeem aantoont dat het grootste deel van de geïdentifi ceerde vereisten inderdaad geïmplementeerd kunnen worden. CMIFed demonstreert ook een aantal manieren om onder andere temporele en spatiele relaties te visualiseren.

Toepassingen

Het in dit proefschrift beschreven werk heeft bijgedragen aan twee grote internationale samenwerkingsverbanden. Het beschreven model is van sterke inv-

loed geweest op de ontwikkeling van SMIL, de Synchronised Multimedia Integration Language. SMIL is een leverancier-onafhankelijke taal om multimedia documenten te beschrijven. Deze taal is ontwikkeld door een werkgroep van het World Wide Web Consortium, waarvan ook leden van de CWI multimedia-groep deel uitmaken. SMIL is bedoeld voor wereldwijd gebruik op het Web, en daarom van groot belang voor de toekomst van multimedia toepassingen. Momenteel worden door belangrijke software fabrikanten browsers ontwikkeld waarmee SMIL documenten via het Web kunnen worden afgespeeld.

Het werk aan het auteursysteem wordt voortgezet als deel van het Chameleon project, ESPRIT-IV project 20597. Eén van de doelstellingen van het Chameleon project is het (semi)-automatisch genereren van multimedia presentaties voor verschillende uitvoerapparaten op basis van één enkel bron-document. Dit wordt mogelijk gemaakt door het CMIFed document format, wat gebaseerd is op het AHM. Hierdoor kunnen op tamelijk eenvoudige wijze presentaties vertaald worden naar de formats voor de verschillende apparaten.

Samenvatting