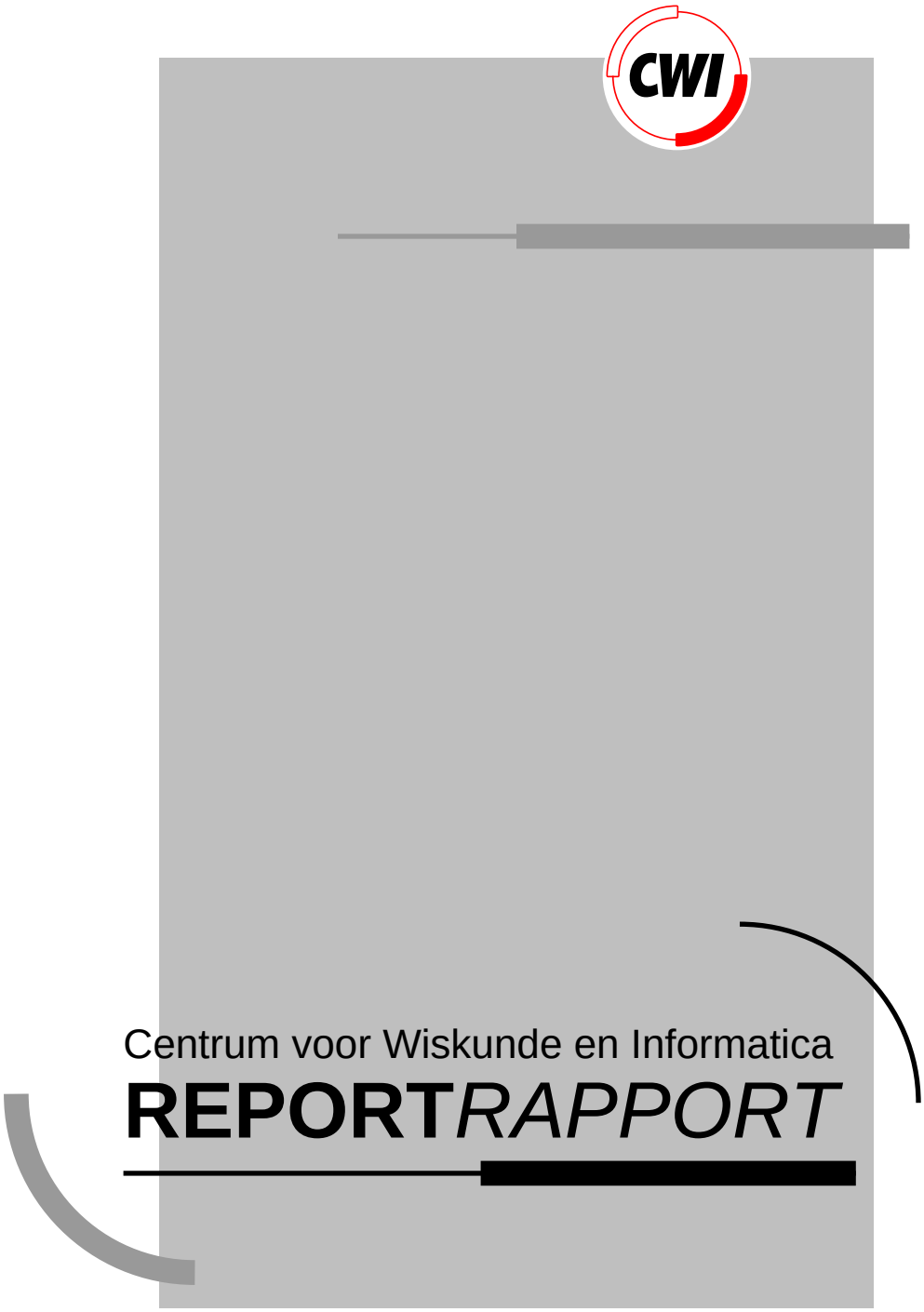


Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

The Amsterdam hypermedia model: extending hypertext to support *real* multimedia

L. Hardman, D.C.A. Bulterman, G. van Rossum

Computer Science/Department of Computer Systems and Telematics

CS-R9306 1993

The Amsterdam Hypermedia Model: extending hypertext to support *real* multimedia

Lynda Hardman, Dick C. A. Bulterman, Guido van Rossum

CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Email: Lynda.Hardman@cwi.nl

Abstract

A model of hypermedia is presented that allows the combination of "hyper-structured" information with dynamic multimedia information. The model is derived by extending the Dexter hypertext reference model and the CMIF multimedia model. The Amsterdam hypermedia model allows the following, in addition to the model provided by Dexter:

- the composition of multiple dynamic media, in order to specify a collection of time-based media making up a complete multimedia presentation;
- the definition of "channels" for specifying default presentation information, allowing the specification of the presentation characteristics of nodes at a more general level than that for an individual node;
- the composition of existing presentations into larger presentations, taking into account possible clashes of resource usage;
- the inclusion of temporal relations while maintaining the separation of structure and presentation information, where time-based relationships are treated as presentation information;
- the specification of "context" for anchors, in order to determine whether on following a link a part, or the whole, of the presentation is replaced by the information at the destination of the link.

The Amsterdam hypermedia model enables the description of structured multimedia documents, incorporating time at a fundamental level, and extending the hypertext notion of links to time-based media and compositions of different media.

The paper is organised as follows. The Dexter hypertext model and the CMIF multimedia model are summarised, and their limitations for use as a more general hypermedia model are discussed. The extensions included in the Amsterdam hypermedia model are described and a summary of the resulting model is given.

CR Subject Classification (1991): H.5.1, H.5.2, I.7.2
Keywords and Phrases: Hypermedia model, Multimedia, Composition, Channels, Context for Anchors, Synchronization
Note: This report has been submitted to the Hypermedia Journal.

1. Introduction

Systems for authoring and reading hypertext material have existed for a number of years. Each of these systems embodies its own model of hypertext. The Dexter hypertext reference model [Halasz and Schwartz 90] is one of the first formal models of hypertext. The Dexter model describes the structures needed for hyper-links. A drawback of this model, however, is that while it accommodates the inclusion of dynamic data items (information which relies on time for its presentation) it does not include time at the structuring level of *documents*[1]. A model

for *hypermedia* (structured multimedia) information requires the inclusion of dynamic media at a fundamental level.

The multimedia group at CWI has developed the CMIF multimedia model [Bulterman et al. 91] and a prototype authoring system for multimedia documents based on it. The model places emphasis on time and combining different media in a continuous presentation. Important parts of the model are that documents are manipulated by the author at the structural level, they are (dynamically) interpreted into a time-based representation, and channels are used for high level specification of presentation information. The model does not, however, allow the expression of hyper-links within or among documents.

While the Dexter and CMIF models make important contributions, both fall short of providing a general hypermedia model. In our view, such a model needs to describe structured, dynamic information using multiple media. 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. These presentations should not be limited to information on stand-alone workstations, but may refer to documents and data sources on local and remote networks.

The Amsterdam hypermedia model uses the Dexter and CMIF models as a basis for the definition of a hypermedia model. The Amsterdam model allows the creation of identifiable multimedia sequences, the specification of presentation styles for multimedia, the combination of existing multimedia sequences into larger presentations, the separation of structural information from presentation information, and the specification of context. While the Amsterdam model make additions to the Dexter model one of the goals is to introduce the required functionality while limiting the changes as much as possible.

The HyTime standard [Newcomb et al. 91] provides a means of expressing hypermedia information in a standard language, however, it does not specify *what* information is required for describing hypermedia. The Amsterdam hypermedia model gives a high-level description of the structures needed for hypermedia, which, once created, can be expressed in HyTime. (A direct analogy is with creating hypertext information conforming to the Dexter model and expressing it in SGML [Goldfarb 90].)

We describe the Amsterdam hypermedia model in this paper. The following section gives a description of the Dexter (hypertext) and CMIF (multimedia) models and lists their drawbacks for use as a general hypermedia model. Section 3 argues for and describes the extensions incorporated in the Amsterdam hypermedia model. Section 4 summarises the proposed model and highlights the differences with the Dexter model.

2. Limitations of existing hypertext and multimedia models

The Dexter model was created in order to describe hypertext and the CMIF model was developed to describe multimedia. Both models have their advantages and drawbacks for describing hypermedia. We show that the ideas from both models can be taken and built upon to form a model of hypermedia.

Section 2.1 gives a summary of the Dexter model, along with its advantages and drawbacks for use as a hypermedia model. Section 2.2 treats the CMIF model similarly. Section 2.3 briefly lists the extensions required to these models for providing a sufficiently rich model for describing hypermedia. These extensions are further discussed in Section 3.

2.1. Dexter hypertext model

Description

A widely cited hypertext model is the Dexter hypertext reference model [Halasz and Schwartz 90]. This was developed by implementors of different hypertext systems in order to rationalise and make explicit the concepts embedded in existing hypertext systems. A brief description of the Dexter model is given here, with emphasis on the aspects most relevant to this paper. Readers familiar with the model can skip to the following section.

1. "Document" is used to refer to a composite (or possibly atomic) component. This should be thought of as a stand-alone presentation which can be played from beginning to end, or can be jumped to or from by the means of links. If it were only text or graphics it could be thought of as a book, chapter, or section.

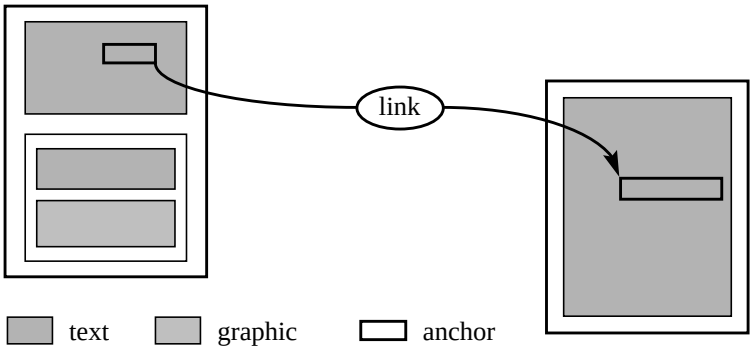


Figure 1. Dexter components

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

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 components 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 discussed in this paper.

The Dexter model introduces the concepts *component* (both *atomic* and *composite*), *link* and *anchor.* Atomic and composite components (often called nodes in system implementations) are related to each other via links. The anchors specify the location of the ends of the links. This is shown schematically in figure 1.

An atomic component contains 3 main parts—content, attributes and presentation specification.

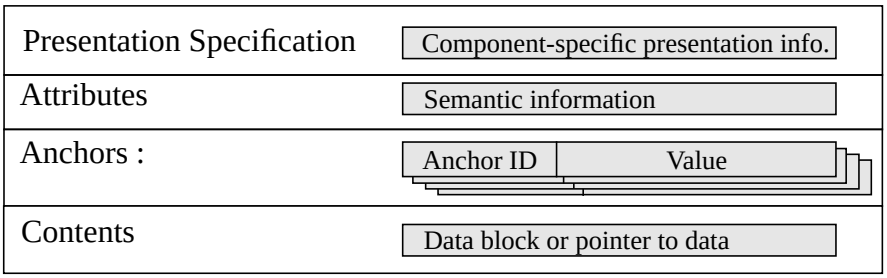
- The *content* is a piece of data of a single medium which can be displayed (or played) such as text, graphics, video, or program fragment.
- The *attributes* allow a semantic description of the component to be recorded. (How this semantic information is derived is outwith the scope of the model.)
- The *presentation specification* holds a description of how the component should be displayed by the system, which may depend on the path taken to reach the component. The presentation specification is independent of the semantic description held in the attributes (although default presentation characteristics, specified separately from the hypertext information, could depend on the semantic description).

Composite components are collections of other components (atomic or composite) and links, which can then be treated as a single component. This hierarchy of components is restricted to a directed, acyclic graph. Each component (atomic or composite) has its own content, attributes, and presentation specification. Anchors specify a part of a component (atomic or composite). For example, within a text fragment the anchor could be a sequence of characters, or within a diagram the anchor could be a graphical object. Links are created between anchors specified in the components. A link consists of a list of anchor references, each with its own direction and presentation specification. (This allows the expression of both simple one source, one destination, uni-directional links, and multiple source, multiple destination, bi-directional links.) A link refers to an anchor via an identifier for the component and an anchor identifier within the component. A component does not refer to links, and has only a list of its own anchors.

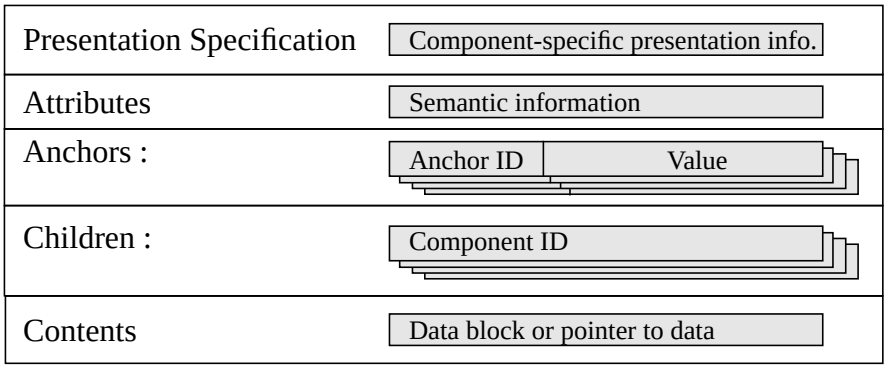
Figure 2 shows a representation of the data structures for (a) the atomic and (b) composite components. Each component has its own unique identifier — this is not shown in the figures.

Limitations

The Dexter model allows the composition of hierarchical structures, and the specification of links between any two components. The deficiencies of the model for hypermedia are that it has no notion of time beyond the within-component layer, no way of specifying higher-level presentation information, and no notion of context for an anchor.



(a) atomic component



(b) composite component

Figure 2. Dexter model

2.2. CMIF multimedia model

Description

The CWI Multimedia Interchange Format (CMIF) model [Bulterman et al. 91], describes a model for representing and manipulating multimedia documents. The CMIF model concentrates on the time-based organisation of information, and the construction of larger documents out of smaller documents (hierarchical structuring). Media objects in the model have contents (which can be played) and synchronization constraints (which are used to specify architecture-independent timing information among the objects). The resulting document can be played as if it were a video sequence — the reader has controls for playing, stopping, pausing etc. the presentation.

The CMIF model includes the following.

- *Data block* contains data of an atomic medium (similar to the content of the Dexter atomic component). This is the smallest unit that can be mapped onto a channel for presentation.
- *Channel*, 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 data from a data block is via a channel. The number of channels within a document is not restricted.
- *Synchronization arcs* are used for specifying timing constraints between data blocks independently of the platform the document is created on.
- *Data descriptor*, a set of attributes describing the semantics of the data block (similar to the attributes in the Dexter model).
- *Event descriptor*, a set of attributes describing the presentation of the data block (similar to the presentation specification in the Dexter model).

There are two important ways of viewing a CMIF document when authoring — the hierarchical view (figure 3) and the channel view (figure 4). The hierarchical view, figure 3(a), shows the document as a structured collection of data blocks to be played in series or in parallel. The structuring of the data blocks in the hierarchy view gives implicit synchronization information: blocks are played either in parallel or serially, as shown in figure 3(b). The

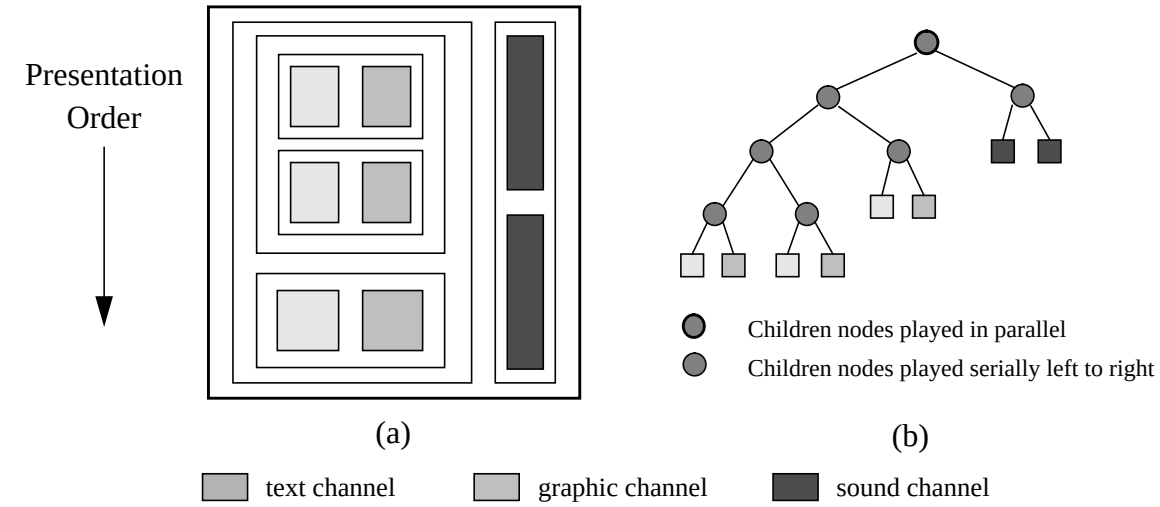


Figure 3. CMIF hierarchy view

(a) The CMIF hierarchy view. Each filled box represents a data block. The rectangles enclosing the boxes represent the hierarchical structuring of the document. The outermost rectangle is the root of the tree. A data block is played before the block beneath it. Data blocks next to each other are started in parallel (unless otherwise constrained by explicit synchronization arcs). The shading of the data blocks indicates which channel the data block uses.
(b) A tree view of the structure in (a).

---

channel view, figure 4, displays the data blocks mapped onto the channels, thus showing the synchronization information, and allows the specification of further explicit synchronization constraints. (For readers familiar with the CWI work, these figures are representations of the structures and not the actual presentations used in the system.)

Limitations

The CMIF model allows the composition of static and dynamic media into time-dependent presentations. Synchronization constraints allow the same document to be played on a variety of architectures while retaining as much as possible of the spirit of the author's original intentions. Channels allow a high-level definition of presentation information. The drawbacks of the CMIF model are that it does not include hyper-links, and that the hierarchical structuring implicitly imposes a particular form of time-based constraint (playing in parallel or serially).

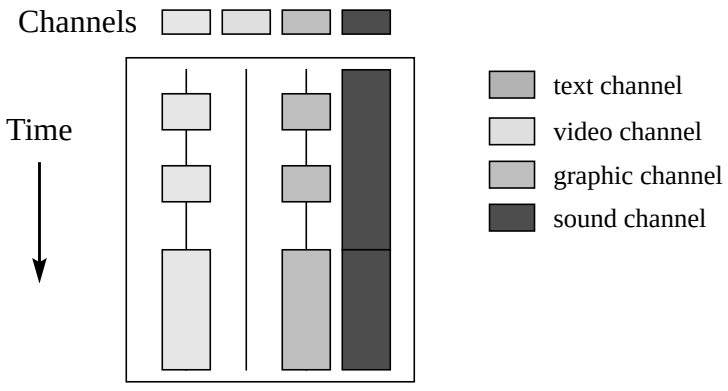


Figure 4. CMIF channel view

The document is the same as that shown in Figure 3. Each filled box represents a Data Block. The length of the block represents the time it takes to play.

2.3. Extensions required for a model of hypermedia

The Dexter hypertext model and the CMIF multimedia model together form a sound base for forming a model of hypermedia. The following list states the extensions needed to the Dexter model to provide a model for hypermedia.

Composition with multiple, dynamic media

A hypermedia model is needed which allows the grouping of items into a presentation and the expression of timing constraints between these items. In hypertext, following a link takes the reader to a destination specified by an anchor. The source anchor is in one window, and the destination anchor is elsewhere in that window, or in a different window. In hypermedia a link can take the reader to a destination consisting of a number of items of (possibly different) static and dynamic media grouped together to form a complete presentation.

Although the Dexter model defines composition, it is inadequate for hypermedia because no account is taken of timing relations between items being grouped into a composite component. It is for this reason that composition in the Dexter model needs to be extended.

Higher level presentation specification

A hypermedia model needs to accommodate sets of presentation specifications applicable to a number of objects. When incorporating items whose presentation specification is specified per object into a complete presentation, the author needs to specify for each item where it is displayed on the screen. If the author then later wants to change parts of the presentation then the presentation specifications in all the affected items need to be changed individually. This is similar to the problem that still exists in less sophisticated word processors where an author is required to specify the layout style individually for every non-standard textline (e.g. captions, or sections headings). For other media types the volume for a number of sounds, or a playback speed for a number of video clips, might be stored.

Attributes for individual data nodes should be recorded separately from those for the presentation style for a number of nodes. The channels from the CMIF model allow the specification of suitable structures.

Combining composite components

A model of hypermedia needs to allow the expression of information required for combining smaller presentations into larger presentations. Composition is easily done in theory, but when authoring in practice there are limited resources in both screen real estate and audio output devices. An author needs to know whether two groups of components can be combined to give a presentation which will remain playable using the available resources. This requires both the ability to compose multiple, dynamic media and is made simpler by the use of higher level presentation specifications.

Temporal relations

A hypermedia model needs to express time-based relations, but these should be treated as presentation information, and kept separate from the link-based structure information. Often, when hypertext systems are extended to handle dynamic media, these media tend to be handled only as “leaf” nodes in the hypertext structure. The component itself contains a time-based medium, but the underlying structure does not represent time. Time needs to be integrated at a more fundamental level in a hypermedia model.

Some hypertext systems use the existing link structures for expressing timing relations. These link structures should not be corrupted by extending then to record time-based information. Combining structure and timing information makes maintenance of both these types of information tedious for the author. A model of hypermedia needs to express time-based relations separate from the structure information.

Context for anchors

In a model of hypermedia some way is needed for specifying when readers follow a link whether they are taken to a completely new presentation, or whether only a part of the current presentation is replaced.When following a link in a typical hypertext system the reader is taken from the current document to a new one (or perhaps to a different place in the same document). In hypermedia it is useful for the author, however, to be able to specify which

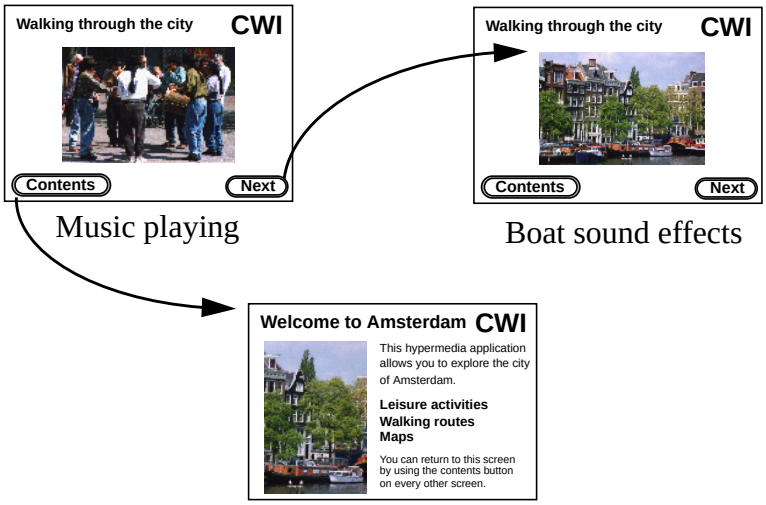


Figure 5. A typical multimedia document

The "Next" button takes the reader to the same section where the boat picture and sound effects replace only the musicians picture and music. The "Contents" button replaces the complete presentation with the contents screen of the Amsterdam tour.

---

parts of the presentation should remain and which should be replaced when a link is followed. This gives the feeling of remaining in the current section, or changing section to a completely different part of the application.

For example, in figure 5 the anchor for the "Next" button in the left-hand presentation has as context the musician-related parts of the presentation, whereas the anchor for the "Contents" button has as context the complete presentation.

3. The Amsterdam hypermedia model

In this section we present the Amsterdam hypermedia model which meets the requirements stated in the previous section. For each of the extensions listed in section 2.3 we discuss its motivation, propose a solution and give a summary of the implications for the proposed hypermedia model. A summary of the complete model is given in section 4.

3.1. Composition with multiple, dynamic media

A hypermedia model is needed which allows the grouping of items into a presentation and the expression of timing constraints between these items. This deficiency in existing models has already been encountered by a group at Brown University's Institute for Research in Information and Scholarship (IRIS), who were investigating the issues involved with creating structured documents with dynamic media [Palaniappan et al. 90]. The group considers the construction of a set of multimedia documents through which the reader can browse. The article raises the question of following a link to multiple anchors. The example given is that the author might want to show a videodisk sequence of paintings of Napoleon's retreat, while playing part of the 1812 Overture. The article describes a method of grouping anchors together so that when the reader follows a link multiple destination anchors are displayed. There is no possibility, however, of defining how the nodes containing these anchors can be synchronized with respect to each other.

Timing relations

When grouping components into a presentation, timing constraints among the children of the composite need to be specified. These can be specified with respect to the parent component or with respect to other sibling components. The duration of the parent component is dependent on the durations of its children and the timing relations among them. Figure 6 shows four synchronization possibilities with respect to the parent component.

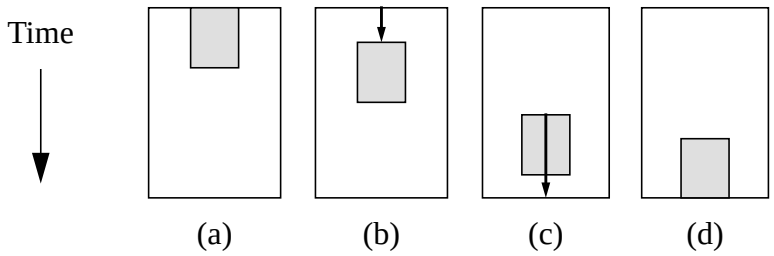


Figure 6. Timing relative to parent

(a) Child starts when parent starts.
(b) Child starts specified delay after parent starts.
(c) Child starts specified time before parent ends.
(d) Child ends when parent ends.

Figure 7 shows synchronization between two children of a composite component, in four of the possible cases.

Composition

A slight change to composition in the Dexter model is the removal of the contents for a composite component. We restrict the presence of contents (data blocks in the CMIF model) to atomic components. This gives a cleaner notion of composition and removes complications about where the contents exist in the presentation hierarchy—they are always at the leaf nodes.

In the Amsterdam model, a composite component specifies the children comprising the composite. These may be atomic components (of static or dynamic medium) or composite components—but see the description of the problems with combining composites in section 3.3 “Combining composite components”. The children are identified via unique identifiers. The timing information for each child is given by synchronization relations with the enclosing parent or with other sibling components.

Composition in the Amsterdam hypermedia model is extended from that in the Dexter model by adding timing relations between components in a composite.

3.2. Higher level presentation specification (channels)

Channels are abstract output devices for playing multimedia nodes and are an important part of the CMIF model for describing multimedia documents. (See [Bulterman et al. 91] for further detailed information on Channels.)

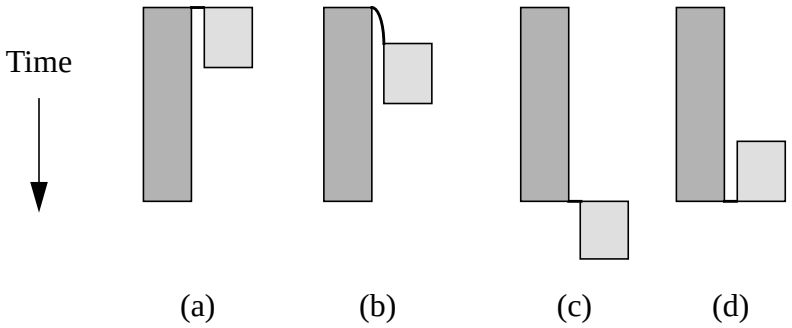


Figure 7. Timing relative to sibling

(a) Children start at the same time.
(b) Right-hand child starts specified delay after left-hand child.
(c) Right-hand child starts after left-hand item ends.
(d) Children end at same time.

Channels are needed for multimedia in the same way that word processors allow the definition of global styles for different structures (for example, section heading or paragraph body). This allows the presentation to be dependent on the structure, and thus different presentations can be applied to the document by specifying a global presentation and not by changing the presentation of every item. Just as in paper documents this encourages consistency throughout the presentation. Flexibility is maintained by allowing local overrides.

Similar to the use of text styles in paper documents, the user can also associate a purpose with a channel in multimedia. For example in the upper half of figure 5 the heading is top left, the main item of interest is the picture in the middle, and links to other places in the document are at the bottom of the window.

The channel defines default presentation characteristics for the medium using the channel. Examples of channel definitions for text would be a default font, size and style and the position and size of the window. A channel for video would be a window and an associated colour map. Overlapping screen-based channels require relative “Z” positions, i.e. the layering order of the windows.

A typical use of channels is to combine several of them into a reusable layout. For example in figure 5 the layout has been reused, where a boat picture replaces that of the musicians and another sound is played. The author is not restricted to the number of channels that can be used, but typically, a number of standard layouts will be designed and re-used, and a few will be used only occasionally.

In the Amsterdam hypermedia model, channels are used to define a default presentation style for the atomic components which are played via that channel. Channels are defined globally (for a collection of documents) and are referenced by each atomic component. A channel applies to only one media type. A number of channels may be appropriate for any one atomic component. Only one atomic component can occupy a channel at any one time.

Channels are part of the presentation specifications in the Dexter model to provide reusable definitions for the various static and dynamic media. They define default presentation information for a particular media type.

3.3. Combining composite components

When combining existing components the resulting composite component has to be playable using the available resources. This may be impossible through overuse of individual channels, or through use of incompatible channels. For example, in figure 8, already existing headings cannot be combined with a full-screen video whose channel uses the whole screen.

When combining two components into a new composite component clashes of resource use need to be calculated. This information can be derived from the channels and timings used by the descendant atomic components, and the synchronization relations between them. This information can then be compared for the two candidate components. The presentation of this information will vary, but figure 9 shows this information for combining the two composites using a variation of the channel view from [Bulterman et al. 91].

Figure 9 shows a straightforward combination. Figure 10 shows a more complex example where the utility of the representation is more apparent. The author can see which parts of the candidate components need to be changed to allow them to be combined. (Neither figure 9 nor figure 10 shows the structural information for the Composite Components.)

Figure 8. Resource overuse

The left hand composite component uses the same screen space as part of the right-hand atomic component, so these cannot be combined.

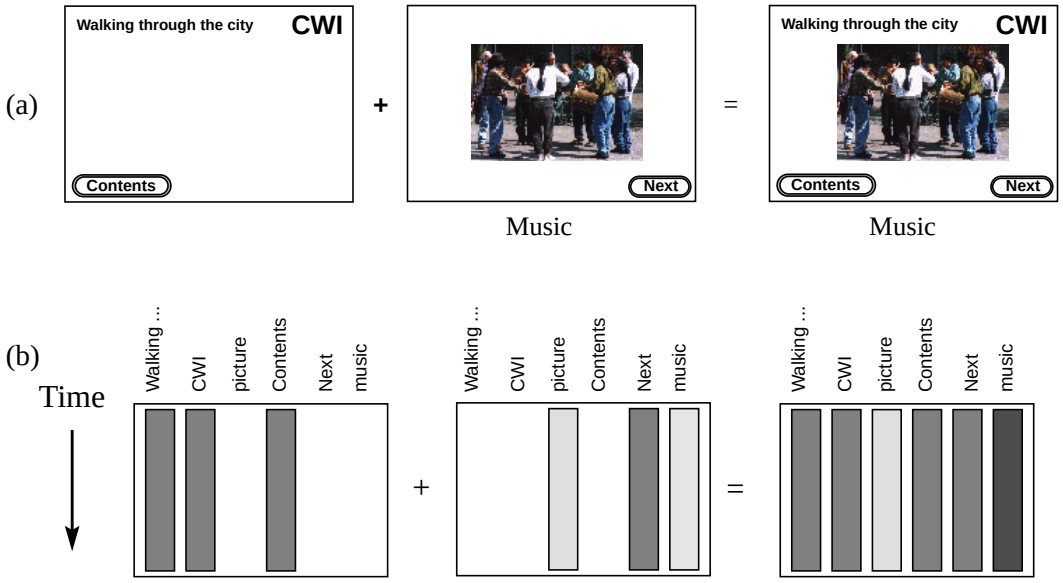


Figure 9. Combining composite components

Combination of two composites is possible.
(a) Screen representation of composites.
(b) Channel representation of composites.

When creating a new composite component from existing components, only the channels themselves need to be compared. Still more powerful is to allow relationships to be defined between channels, so that it is easier to see which channels can be used with each other. Defining groups of channels into layouts is one organising mechanism for enabling this.

Combining composite components does not require an additional extension to the Amsterdam hypermedia model but makes use of the already proposed composition and channel extensions.

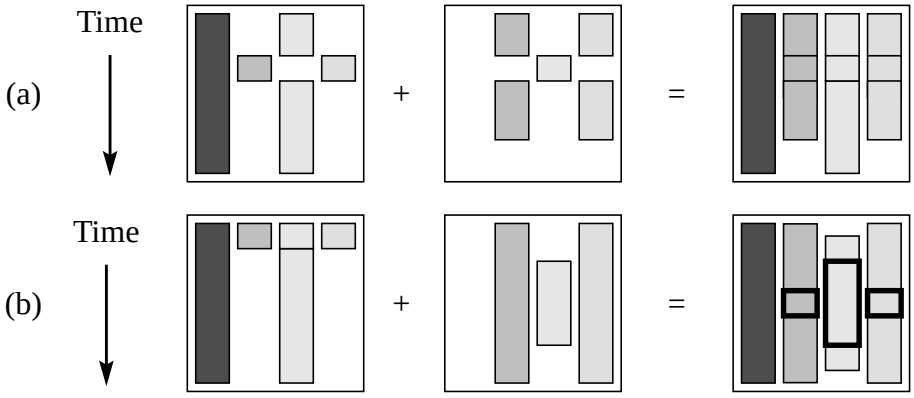


Figure 10. Channel representations of composite components

(a) Two composite components can be combined to form a new, larger composite component.
(b) Combining the candidate components produces double use of the channels in three different places (shown in bold outline).

3.4. Temporal relations

Given the dynamic nature of multimedia data, we need to be able to express temporal relations in a hypermedia model. Two examples of systems that have extended the hypertext model to include dynamic media are Harmony, [Fujikawa et al. 91], and Videobook, [Ogawa et al. 90].

Harmony, built at Osaka University, contains *objects*, similar to Dexter's atomic components. Each object can include *subobjects* (anchors, in Dexter terms) which specify parts of an object and can become the source or destination of a *link*. Relations between objects are represented by links. A link is represented as:
<source-object, conditions, target-object, message-for-target>.
For example:
<dolphin-video, started:30, background-music, play>
specifies that the music should begin playing 30 seconds after the start of the dolphin video. Time constraints are expressed in the Harmony model using this extended link information.

The Videobook system treats (small) multimedia presentations as scenes that can be built up into longer presentations. Timing relationships between the scenes making up the presentation are described within the parent node. Links can be created between scenes: the anchors (called triggers) are defined by scripts specifying the size and duration of an active area on the screen, and the name of the target scene. Timing relationships between non-sibling scenes, or nodes, cannot be specified.

Harmony and Videobook both specify timing relations using the existing link structures. Combining structure and timing information makes maintenance of both these types of information tedious for the author. In the Amsterdam model, time-based relations are expressed as presentation information. This is conceptually cleaner, since it keeps temporal relationships separate from the link-based structure information.

For both time-based and structural information hierarchical and non-hierarchical structures are possible. Our model distinguishes which information is being stored. Hierarchical structuring of a multimedia document is composition, which is discussed in section 3.3 above. Non-hierarchical structuring of a hypermedia document is expressed using links, as specified in the Dexter model.

Timing constraints can be imposed between any two descendants of a composite component. However, it is clearer to make a distinction between timing constraints defined between children of a composite component, and those between any two non-sibling descendants of the composite component. The children of a composite component require timing relations, otherwise neither the author nor the system knows when the items are to be presented. These timing relations are discussed in section 3.1, "Composition of multiple, dynamic media" above.

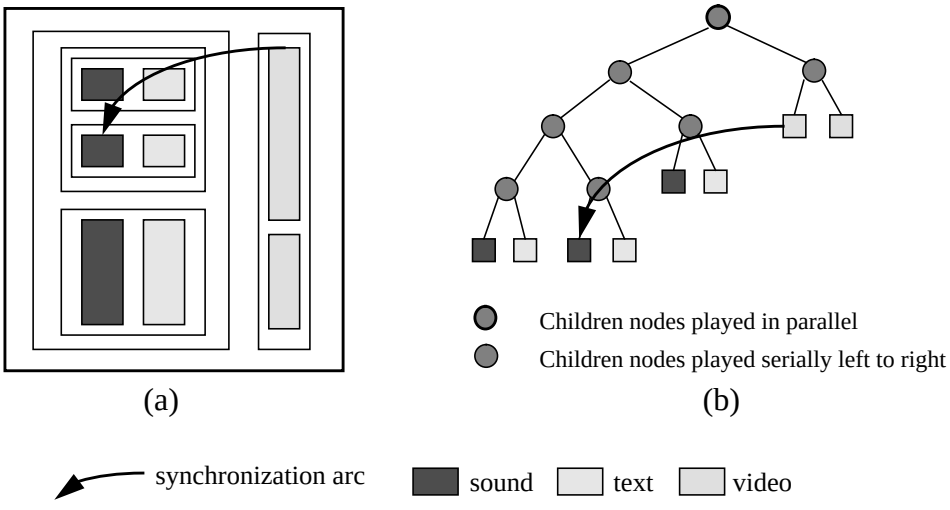


Figure 11. Synchronization arc

The synchronization arc specifies the time delay between the start of the longer video and the beginning of the second part of the sound track. (b) shows a tree view of the structure in (a).

Optional timing relationships between non-sibling descendants are also supported. For example, a longish video fragment is accompanied by a sound track (and corresponding subtitles). The author indicates that the second part of the sound track is to start a specified number of seconds after the video begins. The video fragment and the second part of the sound track are in different levels of the document structure, however the timing dependency is between the two atomic components. An impression of this is given in figure 11. Timing constraints between non-sibling descendants are expressed in the CMIF model using synchronization arcs. These are stored with the whole CMIF document, which becomes, in Dexter terms, a composite component containing both items constrained by the synchronization arc.

The synchronization arcs are relevant only for items within a composite component. If no structural connection exists between two items then there is little point in specifying a presentation relation. The synchronization information is stored in an ancestor (preferably that which is lowest in the hierarchy) composite component of both items. Details on the synchronization arc can be found in [Bulterman et al. 91]. The authoring system should prevent the author from specifying impossible timing constraints (i.e. this is outside the scope of the model).

In the Amsterdam model, hierarchical structure and timing information are expressed via the composite components (described in the previous composition sections, 3.1 and 3.3). Non-hierarchical structure information is expressed in the links and anchors. Non-hierarchical timing information is expressed via the newly introduced structure, the synchronization arcs, stored in a common ancestor of the related items. Dexter specifies both structural types of information, but no explicit timing information (implementation-dependent information can of course be stored in the presentation specification of a component).

3.5. Context for anchors

Current hypertext systems do not make explicit the notion of how much information the reader leaves when following a link. Most systems present a single hypertext node which is either replaced by the target information or is left on the screen in its own window while another window is created to contain the target information. What actually occurs may be determined by the user, the author or, most likely, the way it happens to have been implemented by the hypertext system.

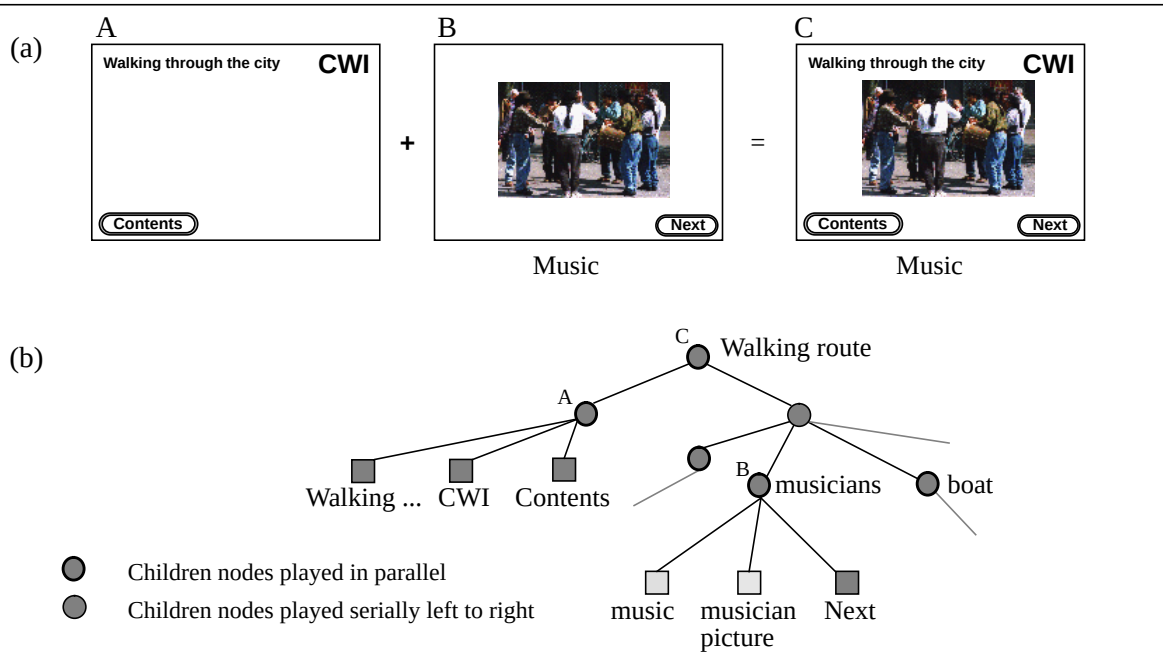


Figure 12. Structure of a presentation

The structure of the presentation in (a) is given in (b). The actual screen layout is shown by C in (a).
The “Contents” anchor is linked to the main contents screen, whereas the “Next” anchor is linked to a boat scene (shown in figure 5). The context for the “Next” anchor is the musicians scene (B), while the context for the “Contents” anchor is the complete walking route scene (C). Following the link from the “Next” anchor leaves the higher-level parts of the document on the screen, whereas activating the “Contents” anchor replaces the whole screen.

Definition The context for an anchor is the part of the document structure which is affected when following a link to or from the anchor.

Context is partially implemented in HyperCard [Apple Computer Inc. 87], although not stated explicitly as such. The context for the anchor is either the current card (the card is replaced while the background remains) or the background (the card and background are replaced). The context for the anchor is specified by whether the button is created on the card or the background. The Guide hypertext system [Brown ??], for both the UNIX and PC platforms, has implementations of different variations of context.

The benefit of specifying context is that only part of the document structure displayed on screen need be affected on following a link. Nodes of the presentation higher in the structural hierarchy remain on the screen while only nodes at the lower levels are replaced. This reduces the authoring burden of repeating the same higher-level structures for different presentations. Figure 12 shows two different levels of a document. When the user activates the "Next" anchor then only the components in part "B" need be replaced and those in "A" remain on the screen. The hierarchy can be as deep as the author wishes.

We thus need to specify two things—a medium-specific definition of the anchor (e.g. a string in text, or an image item in a graphic) and the part of the presentation which will be affected on following the link associated with the anchor.

The hypermedia model needs to be extended to accommodate the required functionality by specifying the context along with an anchor. This can be given by specifying a composite component which is to be replaced on following a link from the anchor. (The Dexter model already includes composite, but this is not used in conjunction with each anchor definition.) This is included in the Amsterdam hypermedia model by specifying the composite component identification (the context) and anchor identification in the link. The anchor is then "de-referenced" to one or more anchors associated with atomic component (which then contain the data-type dependent anchor description).

Once we have established the context associated with an anchor, we can choose different display options associated with following a link. The context can be retained or replaced. If replaced then the target context will be displayed where the source context had been presented (this will require checking mechanisms in the authoring system to prevent overuse of available resources). If the context is retained then the presentation can continue playing, or it can pause. The target context then requires extra resources (most likely an additional window) in order to be displayed.

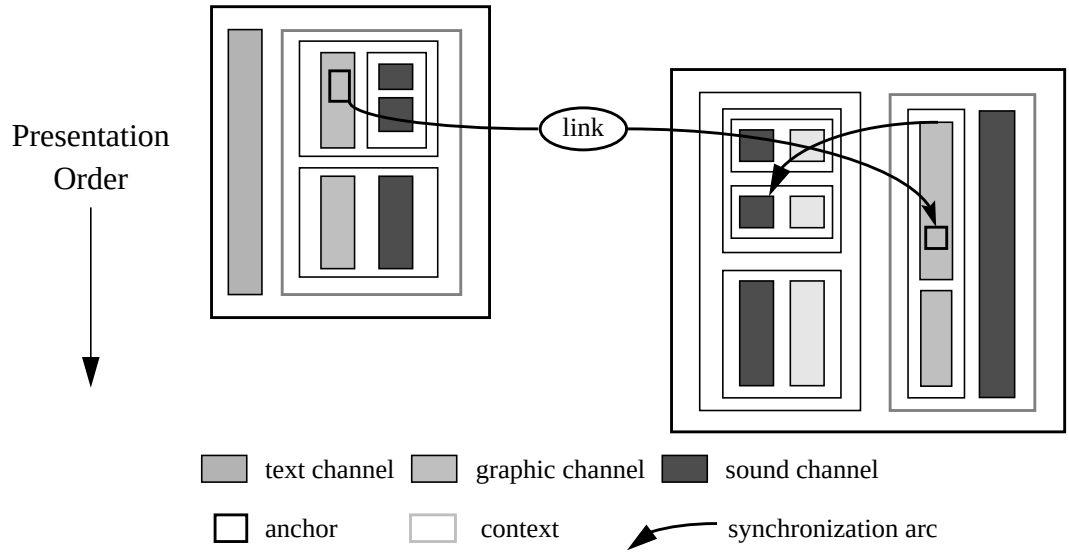


Figure 13. Amsterdam Hypermedia Model components and link

Two Composite nodes (multimedia presentations) connected by a Link. The Link has source and destination Anchors each with their own context. A synchronization arc gives timing constraints within the one composite.
Compare with the Dexter model shown in figure 1.

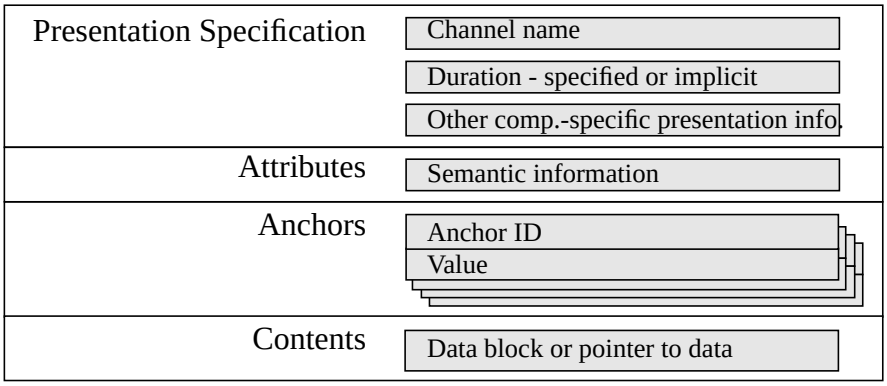
Context is not specified in Dexter, since anchors are defined only in relation to a single node (although that node may itself be composite). The Amsterdam hypermedia model specifies context for an anchor, and does this by making use of the already existing composite items.

4. Summary of the Amsterdam hypermedia model

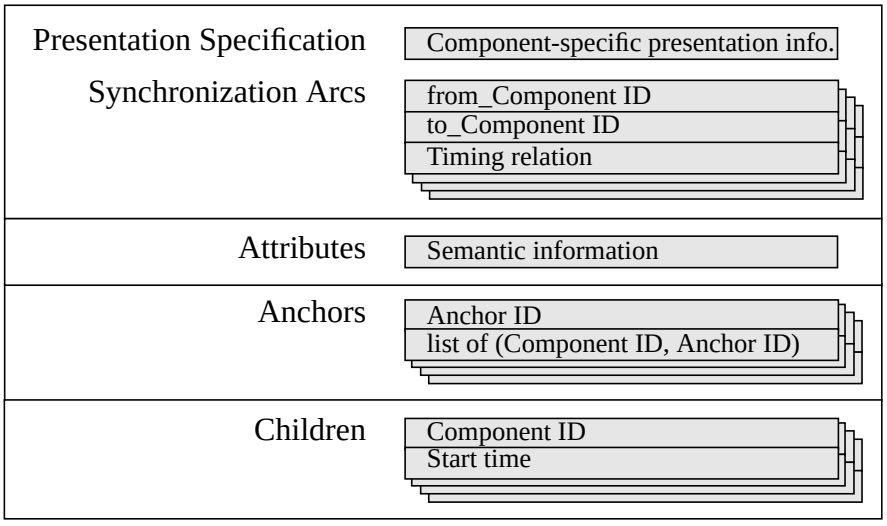
The previous sections describe the extensions required to the Dexter hypertext reference model and the CMIF multimedia model to give a model suitable for describing hypermedia. This section gives a brief summary of how these are integrated into the proposed Amsterdam hypermedia model and how these differ from the Dexter model. Except where explicitly stated, the terms used to describe the Amsterdam model are the same as those in the Dexter model.

The Amsterdam hypermedia model requires a database of atomic components, composite components, links and channels. The channels are an addition to the Dexter model and can be thought of as predefined presentation specifications.

Figure 13 gives an impression of a link between two composite components. This figure should be compared with figure 1, which shows a similar representation for the Dexter model. In figure 13 each composite node is a multimedia presentation. The link has source and destination anchors, and each anchor has a related context (shown with a broken line). The different media are assigned to channels (running vertically).



(a) atomic component



(b) composite component

Figure 14. Amsterdam Hypermedia Model

Compare with the Dexter model shown in figure 2.

Figure 14 shows the conceptual data structure for (a) atomic and (b) composite components. This should be compared with figure 2, which shows a similar representation for the Dexter model. Additions to the atomic component are the channel name and duration as sub-divisions of the presentation specification. Additions to the composite component are the dereferencing of anchors to a list of <Component ID, Anchor ID> pairs, and the start times (or offsets) for the children of the composite. The contents in the composite has been removed, with the implication that only leaf nodes of the structure have related data blocks.

5. Conclusions

Current hypertext models are not sufficient for describing hypermedia. The introduction of time-based media and the composition of multiple nodes of different media require extensions to existing models. This paper describes the extensions required to the Dexter hypertext reference model and the CMIF multimedia model, and presents the Amsterdam hypermedia model.

The Amsterdam hypermedia model emphasizes the importance of composition, and makes use of this for building up structured presentations, while maintaining a way of specifying the presentation relationships between the component items through the use of channels. The ability to describe temporal relations has been included, while ensuring that these are maintained separately from the structural information. Context is introduced as another important concept in hypermedia and its storage within the model indicated.

The Multimedia group at CWI (Centre for Mathematics and Computer Science in Amsterdam) has a working prototype which implements many of the ideas embedded in the presented model. Among these are the ability to author multimedia documents by manipulating their structure, the use of channels, the separation of structure and timing information, and the ability to author simple hyper-links between single nodes (though as yet without context).

The Amsterdam hypermedia model is based on a sound model of hypertext [Halasz & Schwartz 90] and of multimedia [Bulterman et al. 91]. We have argued that the integration of these models forms a basis for a model of hypermedia.

Acknowledgements

We would like to thank the members of the HYPERATE group in the European Community funded DELTA project SAFE (P7061, D1014), in particular Michael Kibby, Willem Bulthuis and Myra Spiliopoulou, and also the employees of OWL, Edinburgh, in particular Stuart Harper, Phil Cooke and Gordon Dougan, all of whom contributed to the ideas in this paper. The concepts expressed in this paper were developed while working with the CMIF editor, implemented by members of the multimedia group at CWI.

References

[Apple Computer Inc. 87] “HyperCard”, Cupertino, CA

[Brown 92] *P J Brown*, “UNIX Guide: lessons from ten years’ development”, ECHT ’92 Nov 30 - Dec 4 1992, Milano, Italy 63 - 70

[Bulterman et al. 91] *Dick C A Bulterman, Guido van Rossum and Robert van Liere*, “A Structure for Transportable, Dynamic Multimedia Documents”, USENIX conference June 1991 Nashville TN, 137 - 155

[Fujikawa et al. 91] *Kazutoshi Fujikawa, Shinji Shimojo, Toshio Matsuura, Shojiro Nishio and Hideo Miyahara*, “Multimedia Presentation System ‘Harmony’ with Temporal and Active Media”, USENIX conference June 1991 Nashville TN, 75 - 93

[Goldfarb 90] *C F Goldfarb*, “The SGML Handbook”, Oxford University Press, 1990.

[Halasz & Schwartz 90] *Frank Halasz and Mayer Schwartz*, “The Dexter Hypertext Reference Model”, NIST Hypertext Standardization Workshop, Gaithersburg, MD, January 16-18 1990

[Newcomb et al. 91] *Steven R Newcomb, Neill A Kipp and Victoria T Newcomb*, "'HyTime' the Hypermedia/Time-based Document Structuring Language", Communications of the ACM, November 1991, 34 (11) 67 - 83

[Ogawa et al. 90] *Ryuichi Ogawa, Hiroaki Harada and Asao Kaneko*, "Scenario-based Hypermedia: A Model and a System", ECHT '90, INRIA, France, Dec 1990 38 - 51

[Palaniappan et al. 90] *Murugappan Palaniappan, Nicole Yankelovich and Mark Sawtelle*, "Linking Active Anchors: a Stage in the Evolution of Hypermedia", Hypermedia 2 (1) 47 - 66