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

We present a model of hypermedia 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 definition of *context* for the source and destination anchors of a link in order to specify the parts of a presentation affected on following 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.

Keywords and Phrases: Hypermedia model, Multimedia, Composition, Channels, Context for Links, Synchronization

1. Introduction

Systems for authoring and reading hypertext material have existed for a number of years. Unfortunately, each of these systems embodies its own model of hypertext. One of the first formal models of hypertext is the Dexter hypertext reference model [1] which describes the structures needed for links among items of information. 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* (“hyper-structured” multimedia) information requires the inclusion of dynamic media at a fundamental level.

Authoring systems for multimedia presentations also exist, again each with their own model of multimedia. The multimedia group at CWI, however, has developed the CMIF multimedia model [2] in order to capture the essence of a multimedia presentation. The group has also built a prototype authoring system, the CMIF editor, based on the model [3]. The CMIF multimedia 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 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.

HyTime [4] provides a means of expressing hypermedia information in a standard language: it does not, however, 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, could be expressed in HyTime. (A direct analogy is with creating hypertext information conforming to the Dexter model and expressing it in SGML [5].)

We describe the Amsterdam hypermedia model in this paper. To put this model in its historical context, 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. It was developed 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.

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.

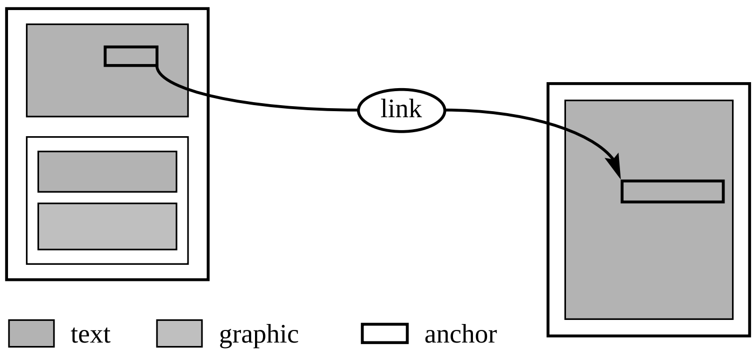


Figure 1. Dexter components

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 Fig. 1.

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

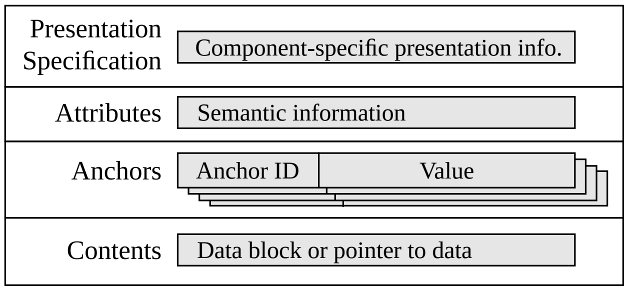
- The *content* is a piece of data of a single medium which can be displayed (or played), such as text, graphics, video, or program fragment.
- The *attributes* allow a semantic description of the component to be recorded. (How this semantic information is derived is beyond the scope of the model.)
- 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.

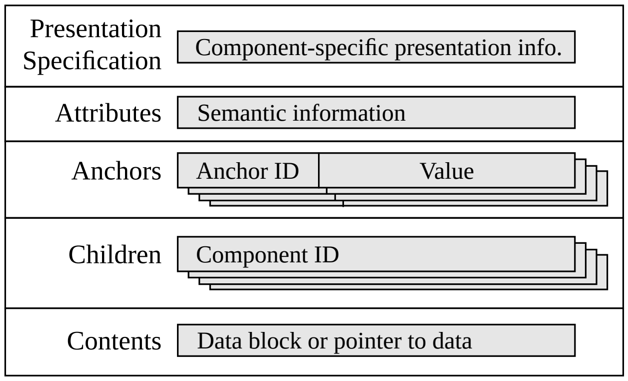
Figure 2 shows a representation of the data structures for (a) the atomic and (b) composite components. Each component has its own unique identifier — this is not shown in the figures.

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. (We discuss context later in the paper.)


Presentation Specification
Component-specific presentation info.
Attributes
Semantic information
Anchors
Anchor ID
Value
Contents
Data block or pointer to data

(a) atomic component


Presentation Specification
Component-specific presentation info.
Attributes
Semantic information
Anchors
Anchor ID
Value
Children
Component ID
Contents
Data block or pointer to data

(b) composite component

Figure 2. Dexter model

2.2. CMIF multimedia model

Description

The CWI Multimedia Interchange Format (CMIF) model, 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. When a document is played the channels are mapped onto physical output devices.
- *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 one instance 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 and the channel view. (Further information about our approach to authoring multimedia is given in [6].) The hierarchical

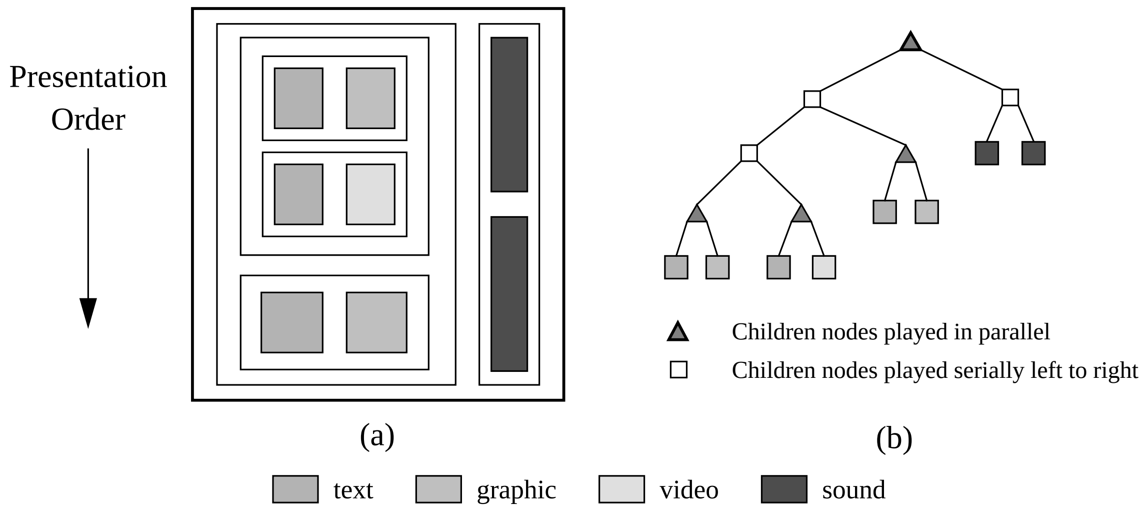

Presentation Order
Children nodes played in parallel
Children nodes played serially left to right
(a)
(b)
text
graphic
video
sound

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, not shown here). The shading of the data blocks indicates which channel the data block uses.
(b) A tree view of the structure in (a).

view, Fig. 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. The channel view, Fig. 4, displays the data blocks mapped onto the channels, thus showing the synchronization information, and allows the specification of further explicit synchronization constraints.

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. Distinguishing serial from parallel composition is a useful authoring paradigm, but it does not increase the theoretical power of the model. A drawback of the CMIF model is that it does not include links.

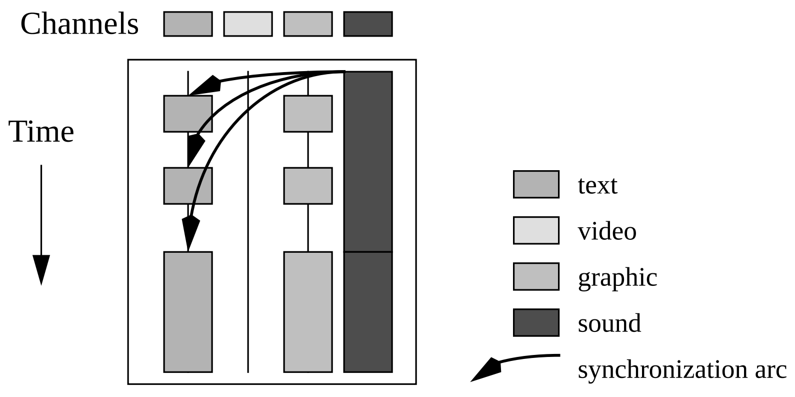


Figure 4. CMIF channel view

Each filled box represents a data block. The length of the block represents the time it takes to play. Timing constraints between the data nodes are derived from the structure shown in Fig. 3, and are further refined by constraints expressed as synchronization arcs.

2.3. Hypermedia model requirements

This section gives an overview of our requirements for model of hypermedia. For each item in the following list we state first a requirement and then which extensions are needed to the Dexter and CMIF models to go towards meeting the requirement. We give more detailed arguments and historical background in the following section.

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 the ability to compose multiple, dynamic media and is simplified 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, time-based issues are included only at the “leaf” nodes in the hypertext structure, for example the Intermedia system [7]. 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, for example the Harmony system described in [8], use the existing link structures for expressing timing relations. However, these link structures should not be corrupted by extending them to record time-based information, since combining structure and timing information makes maintenance of both types of information tedious for the author. A model of hypermedia needs to express time-based relations separate from the structure information.

Context for links

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

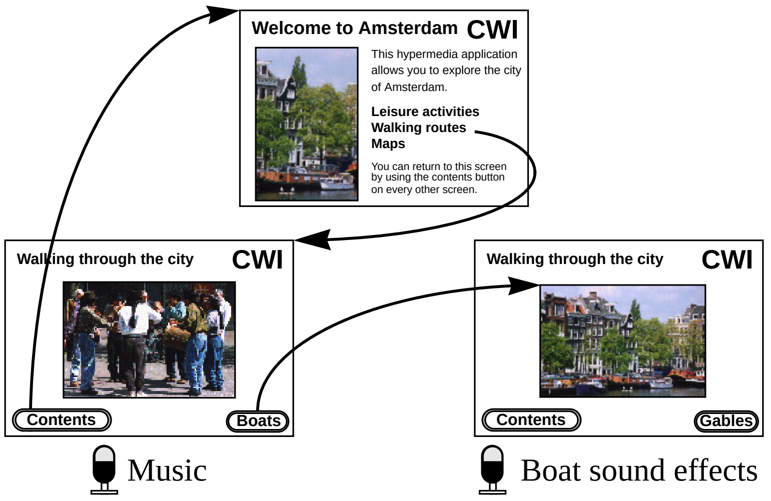


Figure 5. An example hypermedia document

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

---

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 Fig. 5 following the link from the *Boats* button in the left-hand presentation replaces the musician-related parts of the presentation, whereas following the link from the *Contents* button replaces 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 [7]. 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 7

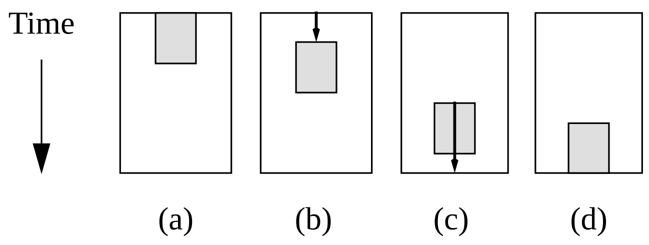


Figure 6. Timing relative to parent

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

shows synchronization between two children of a composite component, in four of the possible cases. This can be extended to multiple siblings by specifying each relation separately.

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 media) or composite components. Composite components can be of two types, parallel or choice, where a *parallel composite component* means that all its children are played, and a *choice composite component* specifies that at most one of its children is played. The intention is that children of a choice component are closely related and are linked to each other. This enables presentations to be built out of sub-presentations—but see the description of the problems with combining composites in section 3.3 "Combining composite components."

The children of a composite component 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. (These relations can be deduced from the parallel composite component structure, or explicitly defined by an author.)

Composition in the Amsterdam hypermedia model is extended from that in the Dexter model by adding timing relations between components in a composite and distinguishing between parallel and choice composite components. Composition is different from the CMIF model in that serial composition has been removed (since it can be expressed using parallel composition and synchronization constraints), and a choice composite component has been introduced.

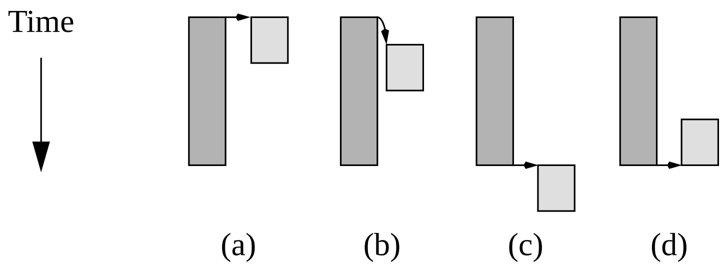


Figure 7. Timing relative to sibling

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

3.2. Higher level presentation specification (channels)

Channels are abstract output devices for playing multimedia items. They are an important part of the CMIF model for describing multimedia documents. (See [2] for further detailed information on channels.)

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 the same document can be presented in different ways by specifying different global presentation styles rather than by changing the presentation of every item. Just as in paper documents this encourages consistency throughout the presentation. Flexibility is maintained by allowing overrides for individual components. Channels are also used as a indication of resource use by the document player.

Similar to the use of text styles in paper documents, the author can also associate a purpose with a channel in multimedia. For example all headings could be played in a heading channel (for example top left of the windows shown in Fig. 5), the main item of interest displayed in large, central channel, and links to other places in the document are displayed in channels 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 Fig. 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 derived from the CMIF multimedia model. They are used to define default presentation information for the various static and dynamic media used in a document. They can be thought of as providing part of the presentation specification defined in the Dexter model.

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 the use of incompatible channels or through the overuse of individual channels. The first case is illustrated in Fig. 8, where existing headings cannot be combined with a video in a channel which uses the whole screen. The second case is illustrated in Fig. 9, where a potential composite component attempts to play multiple atomic components on the same channel simultaneously.

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,

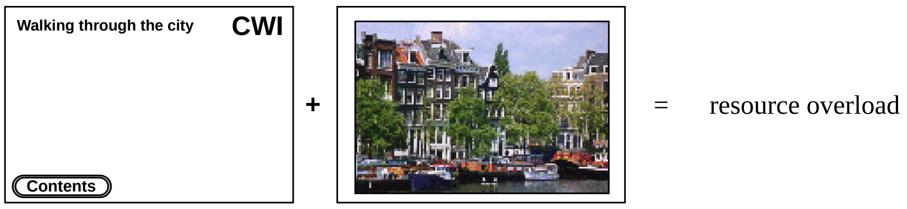


Figure 8. Resource overuse

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

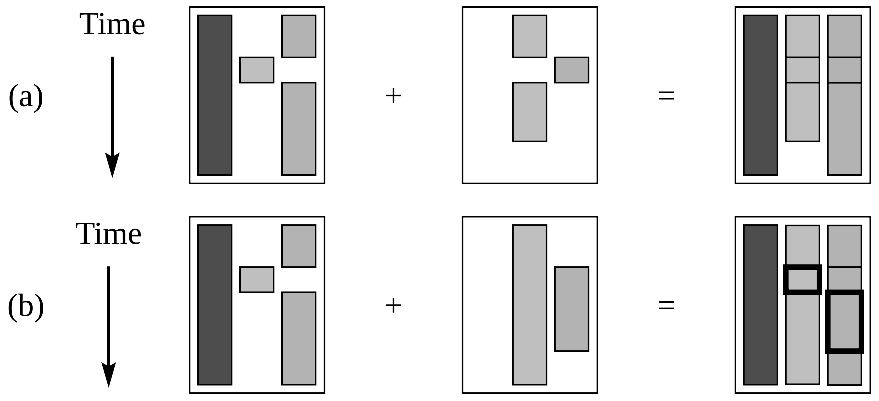


Figure 9. Channel representations of composite components

(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 two different places (shown in bold outline).

and the synchronization relations among them. This information can then be compared for the two candidate components. The presentation of this information will vary, but Fig. 9 shows this information for combining the two composites using a variation of the channel view from the CMIF editor. The author can see which parts of the candidate components need to be changed to allow them to be combined. (Note that Fig. 9 shows no structural information for the composite components.)

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.

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 [8] and Videobook [9].

Harmony, built at Osaka University, contains *object*s, 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.

Buchanan and Zellweger [10] take an object-based approach to specifying timing relations in multimedia, but they do not address the problems associated with using hierarchically structured hypermedia documents.

For both timing and structural information, hierarchical and non-hierarchical relations are possible. Our model distinguishes which information is being stored. We have already dealt with structural information: hierarchical structuring of a multimedia document is composition, which is discussed in section 3.3 above; non-hierarchical structure in 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.

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 Fig. 10. 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 of both items (preferably that which is lowest in the hierarchy). Synchronization arcs define timing relations in terms of minimum and maximum acceptable delays between intervals within a data node. (Further details on synchronization arcs can be found in [2].) The authoring system should prevent the author from specifying impossible timing constraints (i.e. this is beyond 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).

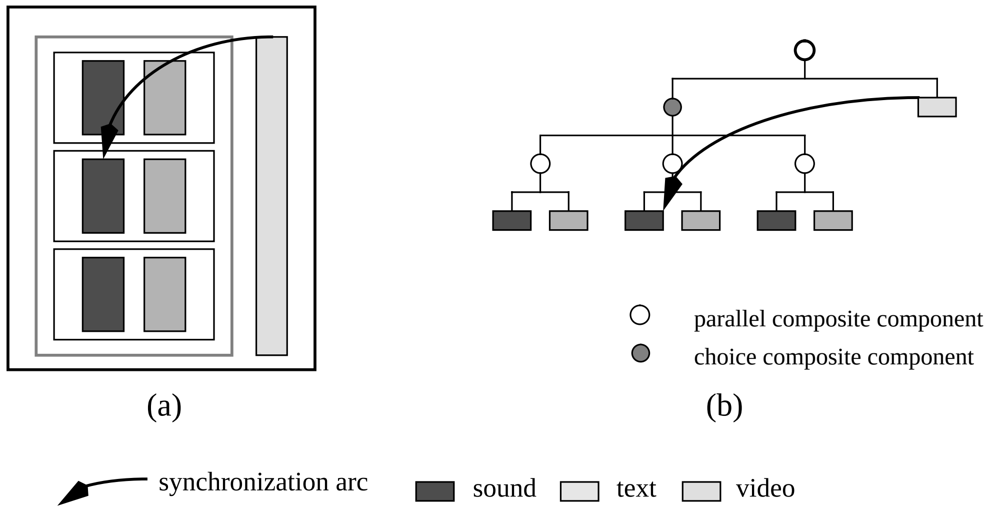


Figure 10. Synchronization arc

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

3.5. Context for links

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 destination information or is left on the screen in its own window while another window is created to contain the destination information, as in NoteCards [11]. What actually occurs may be determined by the author or, most likely, by the way it happens to have been implemented in the hypertext system.

We define the source or destination context for a link as the part of the document structure which is affected when following the link to or from an anchor.

Context can be found in existing hypertext systems, although not stated explicitly as such. For example, the Guide hypertext system [12] for both the UNIX and PC platforms, has implementations of different variations of context. For example, one link type allows the author to specify, via the use of a source region, which of the surrounding material should be replaced on following the link. HyperCard [13] also implements a form of context: following a link may replace only the card, or the card and the background.

A 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. For example when the reader activates the *Boats* anchor in Fig. 11 then only the components in part C are replaced while those in B remain on the screen. The document in the figure uses only two different levels but the model imposes no restriction on the depth of the hierarchy.

For each anchor associated with a link the author needs to specify three things: (a) whether it is a source or destination of the link, (b) a medium-specific definition of the anchor (e.g. a string in text, or a part of the image in a graphic) and (c) the context—the part of the presentation which will be affected on following the link from (or to) the source (or destination) anchor.

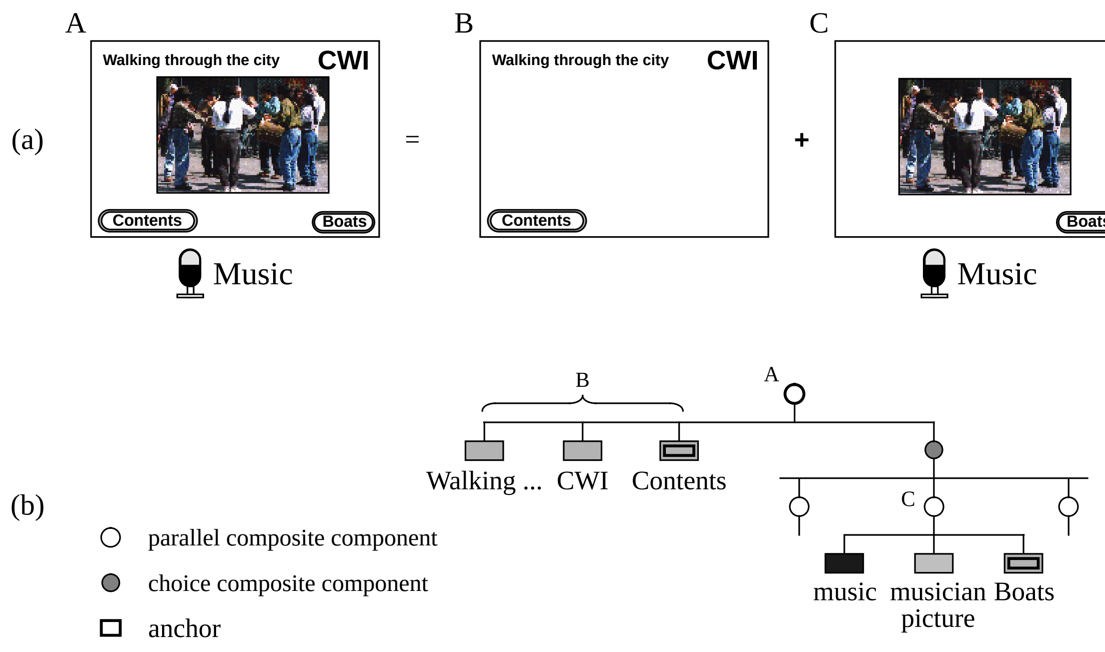


Figure 11. Structure of a presentation

The structure corresponding to the presentation shown in (a) is given in (b). The *Contents* anchor is linked to the main contents screen, and the *Boats* anchor is linked to a boat scene (shown in Fig. 5). The context for the link from the *Contents* anchor is the complete walking route scene (A), so that following the link replaces the whole screen. The context for the link from the *Boats* anchor is the musicians scene (C), so that following the link leaves the higher-level parts of the document on the screen (i.e. those shown in B).

A hypermedia model can accommodate the required functionality by specifying the context along with each of the anchors associated with a link. The context is specified in the Amsterdam hypermedia model by stating a (probably composite) component identification with each anchor. This anchor can reference anchors in descendants of the composite. Eventually the anchors will belong to atomic components (which contain the data-dependent anchor descriptions). These are the areas which can be used on-screen for denoting the end of the link. (The Dexter model already allows the specification of composite components with a link, but these are not used explicitly to define a context.)

Once we have established the context associated with the ends of a link, we can choose different display options associated with following a link. The source context can be retained or replaced. If replaced then the destination context will be displayed where the source context had been presented (this will require checking mechanisms in the authoring system to ensure the destination context can be played in the resources used by the source context). If the source context is retained then the presentation can continue playing, or it can pause. The destination context then requires extra resources (most likely an additional window) in order to be displayed. The display option for the source context also needs to be stored with the link.

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

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.

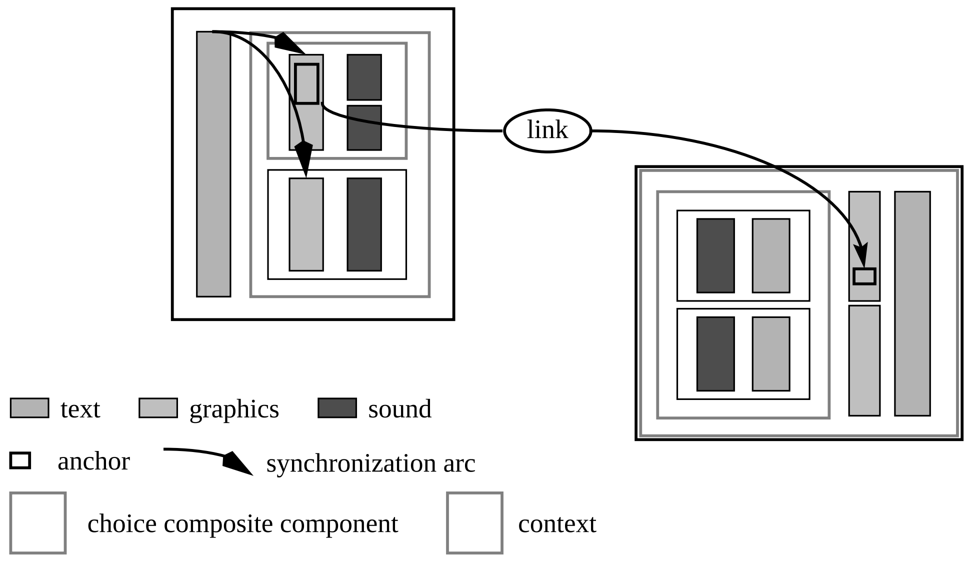
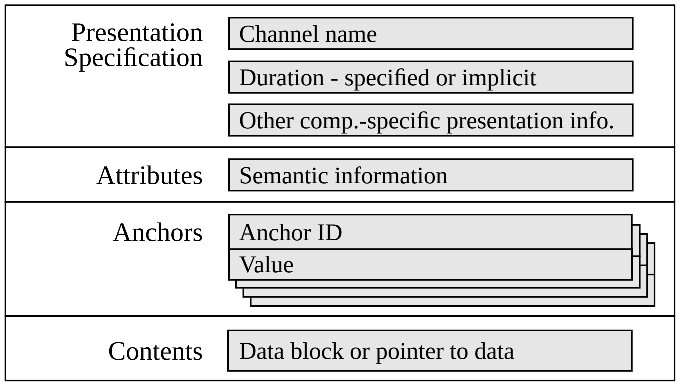
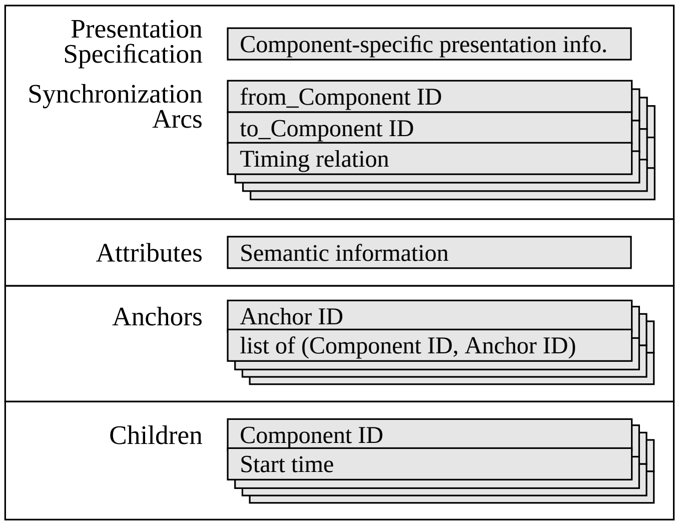


Figure 12. Amsterdam Hypermedia Model components and link

Two composite components (multimedia presentations) are connected by a link. The link has source and destination anchors and contexts. Synchronization arcs give timing constraints between non-sibling components within a composite. This figure can be compared with the Dexter model shown in Fig. 1.


Presentation Specification
Channel name
Duration - specified or implicit
Other comp.-specific presentation info.
Attributes
Semantic information
Anchors
Anchor ID
Value
Contents
Data block or pointer to data

(a) atomic component



(b) composite component

Figure 13. Amsterdam Hypermedia Model

This figure can be compared with the Dexter model shown in Fig. 2:
(a) the presentation specification for an atomic component uses a channel name, and a duration is specified; (b) a composite component has been extended to include synchronization arcs, anchors which can reference anchors of descendants, and a start time for each child component.

Figure 12 gives an impression of a link between two composite components. This figure should be compared with Fig. 1, which shows a similar representation for the Dexter model. In Fig. 12 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).

Figure 13 shows the conceptual data structure for (a) atomic and (b) composite components. This should be compared with Fig. 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 insufficient for describing hypermedia. The introduction of time-based media and the composition of different media require extensions to existing models. Multimedia models, which include at most rudimentary links, are also insufficient for describing hypermedia. However, both include aspects necessary for describing hypermedia. This paper presents the Amsterdam hypermedia model which is based on the Dexter hypertext model and the CMIF multimedia model.

The Amsterdam hypermedia model emphasizes the importance of composition, and makes use of this for building up structured documents, while allowing the specification of presentation specifications of atomic components 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 is indicated.

The Multimedia Kernel Systems group at CWI (Centrum voor Wiskunde en Informatica—Centre for Mathematics and Computer Science) 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 links between single (composite or atomic) nodes.

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 the third author, Jack Jansen and Sjoerd Mullender.

References

[1] HALASZ F. *and* SCHWARTZ M. The Dexter Hypertext Reference Model. *In: NIST Hypertext Standardization Workshop*. Gaithersburg, MD, January 16-18 1990

[2] BULTERMAN D.C.A., VAN ROSSUM G. *and* VAN LIERE R. A Structure for Transportable, Dynamic Multimedia Documents. *In:* Proceedings of the Summer 1991 USENIX Conference. Nashville, Tennesse, 1991, 137 - 155.

[3] VAN ROSSUM G., JANSEN J., MULLENDER K.S. *and* BULTERMAN D.C.A. CMIFed: a Presentation Environment for Portable Hypermedia Documents. *In:* Proceedings of the First International Conference on Multimedia. Anaheim, California, August 1993. (Also available as CWI Report CS-R9305).

[4] NEWCOMB S.R. ,KIPP N.A. *and* NEWOMB V.T. ‘HyTime’ the Hypermedia/Time-based Document Structuring Language. *Communications of the ACM*, 34 (11) November 1991, 67 - 83.

[5] GOLDFARB C.F. *The SGML Handbook*. Oxford University Press, 1990.

[6] HARDMAN L., VAN ROSSUM G. *and* BULTERMAN D.C.A. Structured Multimedia Authoring. *In:* Proceedings of the First International Conference on Multimedia. Anaheim, California, August 1993. (Also available as CWI Report CS-R9304.)

[7] PALANIAPPAN M., YANKELOVICH N. *and* SAWTELLE M. Linking Active Anchors: a Stage in the Evolution of Hypermedia. Hypermedia 2 (1) 1990, 47 - 66.

[8] FUJIKAWA K., SHIMOJO S., MATSUURA T., NISHIO S. *and* MIYAHARA H. Multimedia Presentation System 'Harmony' with Temporal and Active Media. *In:* Proceedings of the Summer 1991 USENIX Conference. Nashville, Tennesse, 1991, 75 - 93.

[9] OGAWA R., HARADA H. *and* KANEKO A. Scenario-based Hypermedia: A Model and a System. *In:* A. Rizk, N. Streitz and J. André, *eds. Hypertext: Concepts, Systems and Applications.* Proceedings of the European Conference on Hypertext. INRIA, France, November 1990, 38 - 51.

[10] BUCHANAN C.M. *and* ZELLWEGER P.T. Specifying Temporal Behavior in Hypermedia Documents. *In:* D. Lucarella, J. Nanard, M. Nanard and P Paolini, *eds. ECHT '92*. Proceedings of the Fourth ACM Conference on Hypertext. Milano, Italy, Nov 30 - Dec 4 1992, 262 - 271.

[11] HALASZ F.G. Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31(7) July 1988, 836 - 852.

[12] BROWN, P.J. UNIX Guide: lessons from ten years' development. *In:* D. Lucarella, J. Nanard, M. Nanard and P Paolini, *eds. ECHT '92.* Proceedings of the Fourth ACM Conference on Hypertext. Milano, Italy, Nov 30 - Dec 4 1992, 63 - 70.

[13] APPLE COMPUTER INC. "HyperCard", Cupertino, CA, 1987.