# SUPPORTING ADAPTIVE AND ADAPTABLE HYPERMEDIA PRESENTATION SEMANTICS

D.C.A. BULTERMAN, L. RUTLEDGE, L. HARDMAN and J. van OSSENBRUGGEN*
*CWI: Centrum voor Wiskunde en Informatica*
*P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*
*{Dick.Bulterman, Lloyd.Rutledge, Lynda.Hardman}@cwi.nl*

**Vrije Universiteit*
*Dept. of Math. and Computer Science*
*De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands*
*jrvosse@cs.vu.nl*

**Abstract.** Having the content of a presentation adapt to the needs, resources and prior activities of a user can be an important benefit of electronic documents. While part of this adaptation is related to the encodings of individual data streams, much of the adaptation can/should be guided by the semantics in and among the objects of the presentation. The semantics involved in having hypermedia presentations adapt can be divided between *adaptive hypermedia*, which adapts autonomously, and *adaptable hypermedia*, which requires presentation-external intervention to be adapted. Understanding adaptive and adaptable hypermedia and the differences between them helps in determining the best manner with which to have a particular hypermedia implementation adapt to the varying circumstances of its presentation. The choice of which type of semantics to represent can affect speed of the database management system processing them. This paper reflects on research and implementation approaches toward both adaptive and adaptable hypermedia and how they apply to specifying the semantics involved in hypermedia authoring and processing. We look at adaptive approaches by considering CMIF and SMIL. The adaptable approaches are represented by the SGML-related collection of formats and the Standard Reference Model (SRM) for IPMS are also reviewed. Based on our experience with both adaptive and adaptable hypermedia, we offer recommendations on how each approach can be supported at the data storage level.

**Keywords:** adaptive hypermedia, adaptable hypermedia, CMIF, SMIL, HyTime, DSSSL, SRM.

## 1 INTRODUCTION

A hypermedia presentation is a structured collection of hypermedia objects. Each object can be considered to be a static element, or it can be an element that individually or in concert with other objects, has its presentation tailored to the needs of the user. Such tailoring can be based on resource constraints (such as bandwidth), but they could also be tailored to the semantic needs of the user.

In this paper *adaptive hypermedia* is hypermedia that adapts autonomously; that is, they can provide alternatives of individual parts of a presentation under varying circumstances based on directives contained in the presentation definition. This contrasts with *adaptable hypermedia,* in which a definition of a single abstract presentation is adapted by use of directives that are external to the base presentation's definition. Both types of adaptation in hypermedia typically account for varying circumstances including user characteristics, system characteristics and intent of the presentation when processed for the user. Different approaches for making hypermedia adapt have been taken by a number of standards and research initiatives. Each approach applies its own distinction between adaptive and adaptable semantics and how they are processed into interaction with the user. Typically adaptable hypermedia is more versatile than adaptive hypermedia but it requires a more complex processing activity to support presentation transformation.

This paper contrasts adaptive and adaptable hypermedia. In section 2, we present an overview of existing approaches to supporting adaptive behavior. In section 3, we consider adaptable hypermedia. In section 4, we consider the relative merits of both approaches and discuss the processing support required for implementing adaptive and adaptable presentations efficiently. Section 5 provides closing comments.

## 2 ADAPTIVE APROACHES

Adaptive presentation control, in which the adaptation occurs autonomously in the context of the source presentation, is the dominant form of adaptation. This section provides background information on current adaptive approaches and then give examples of how two related formats (CMIF and W3C's SMIL) implement adaptive control.

### 2.1 A SURVEY OF ADAPTIVE APPROACHES

The basic problem that adaptive presentations try to solve is the alteration of the presentation's content to meet the needs of the user. At a low level of detail, this involves changing the representation of a piece of data by substituting a low-quality version of an object for a high-quality (and high-bandwidth) version at data access time. This approach typically falls under the heading of *quality of service* adaptivity. It will not be considered in detail in this paper, since the processing of alternatives is not driven by semantics but by syntax.

An intuitive approach to supporting multiple semantic encodings in a presentation is to define separate projections for each composite presentation, each of which could be configured for the alternatives available. If, for example, we wanted to support a multi-lingual presentation, we could generate one complete projection for the Dutch version of the presentation and another one for the English version. This approach is also not treated in detail, since it is an example of simple presentation substitution rather than presentation adaptivity.

There are several ways that adaptive content can be added to a presentation. These fall into two categories: programming-based and declarative. Programming-based control is perhaps the most obvious form of adaptivity control. At its heart is the notion of providing a program or script-based directives within the document that analyze the runtime situation and 'does the right thing' for each data request issued. (In this section, we only consider such processing when all alternative are defined explicitly in the source presentation.) This is the approach taken by HTML-4.0, often called Dynamic HTML [29].

Declarative approaches provide a set of alternatives in which the author states the alternatives available and the conditions under which each alternative is preferred, either directly or indirectly, but not the logic required to implement the choice. A simple form is the definition of explicit alternatives within the contents for each particular media object reference. This is similar to the approach used within HTML for individual media objects:

```
      <img src='anchor.gif' alt="Dick's face">
```

A more complex variant is a complete dynamic creation of the entire presentation at runtime using indirect access to all possible objects stored in a database and then only rendering those objects that are required at a particular moment in the presentation [22]. Here, the database query could include sufficient semantic information so that an appropriate representation (or sub presentation could be built on demand.) A compromise approach is Transparent Content Negotiation (TCN) [17]. Here, the document contains a single reference to an object. At time of access, a process of negotiation takes place that allows the 'right' object to be extracted from the server.

Although many declarative approaches allow semantic processing of alternatives, most practical systems do not make use of this power. It is interesting to compare this approach to resource management with that used in a typical HTML browser. Many browsers give the user the ability to turn images or sounds on or off. This is done to make presentation rendering a positive experience on slow communications lines. What such browsers should probably do is to support a notion of turning off irrelevant images rather than all images. Based on the user's needs, the browser should be able to analyze the content and adjust its rendering accordingly.

## 2.2 CMIF AND CMIF's CHANNEL ARCHITECTURE

One of the original goals of CWI's CMIF project [15] was the definition of an information grouping abstraction that would be useful in specifying collections of related content during authoring, and then selection one or most sets of content at runtime by the user based on that user's needs. The name given to this abstraction was the *Logical Resource Channel*, or simply the *Channel* [9].

The purpose of a channel is to be a grouping abstraction for a set of media items that share some common attributes. These may include physical attributes such as screen position or text color, or they may be logical attributes, such as natural language or presentation priority.

The channel provides a logical thread upon which media objects can be placed. This thread can be turned on or off during the presentation based on the needs of the user or the user's agent (that is, the user interface or the runtime support system). In this way, the Channel abstraction can be used to support user-centered adaptation, but it can also be used by the runtime system to select more traditional QoS adaptation of content alternatives.

CMIF channels have not only a strong logical association among media objects on that channel, they also share presentation rendering and scheduling associations as well. In CMIF, it is not appropriate to speak of *the* audio or video channel—as if there was only a single video output stream—but rather *an* audio or video or text or image channel. An application may have many different text, video, audio, image or control channels, each of which is tailored to a specific logical grouping. Ultimately, the objects on a channel may get directed to a renderer of a particular type, but such a renderer may be responsible for multiple concurrent high-level information streams.

Any media item that is activated needs to be placed somewhere on the screen or an a loudspeaker. When several objects are rendered to the same space, it may make sense to manage this space based on a set of common attributes. Similarly, the actual rendering of a media object needs to be handled by some piece of code or device that is also constrained by a set of common properties. Finally — and most importantly — it may be that a set of media objects can be grouped not only by space and type, but also based on their semantic properties. For example, in a new broadcast, a “Dutch Audio” channel could contain all of the audio in that document that is spoken in Dutch. Alternatively, a “Anchor-Audio-Dutch” channel could be defined that contains the Dutch version of all of audio associated with the anchor. If a “Anchor-Audio-English” channel existed, a user potentially select which language they wanted at runtime.

Figure 1 shows a visual representation of a channel. The channel has a header that contains channel attributes. Each channel supports a particular type of output. This type can range from simple text, through complex composite multimedia data, to non-visible control operations. (A video channel icon is displayed.) Associated with a channel is its virtual timeline; media item activation instance descriptors are placed on this timeline. The timeline is virtual in that it does not show fixed offsets within the presentation: hyperlink and/or timing control (like a loop) in a particular instance will determine a presentation’s ‘real time’. Associated with the virtual timeline is the presentation’s computed time reference line. This timeline is computed by the scheduler based on information available in the CMIF data structure.

The current run-time environment at CWI supports sixteen channel types:

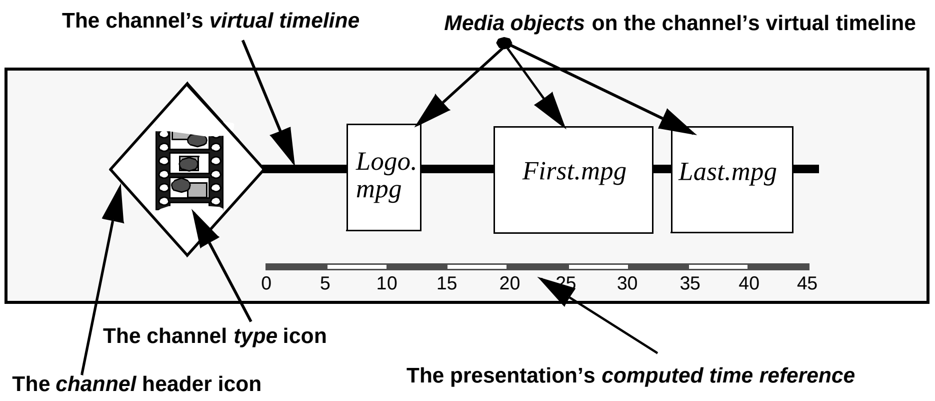

The channel’s *virtual timeline*
*Media objects* on the channel’s virtual timeline
Logo. mpg
First.mpg
Last.mpg
0 5 10 15 20 25 30 35 40 45
The channel *type* icon
The *channel* header icon
The presentation’s *computed time reference*

*Figure 1:* . The CMIF *logical resource channel* visual abstraction

• audio-only: *sound, Midi*
• video-only: *movie, MPEG*
• audio-video: *MPEG*
• text channels: *HTML, text, label*
• illustration: *graph, image*
• background: *layout*
• control: *cmif, Null, Python, Shell, Socket, VCR*

The list above emphasizes the physical properties of the media type, rather than the logical association among groups of generic media objects. This reflects our experience that, from an author's perspective, there are several levels of grouping that need to be managed within an application simultaneously. These are: layout grouping, renderer grouping and semantic grouping.

While the view shown in Fig. 1 is fairly typical for time-line based systems, the combination of multiple virtual timelines into a presentation timeline is less typical. This is shown in Figure 2. Here we see a presentation fragment in which one video channel, two audio channels and two text channels are shown. The runtime projection of this presentation will in all probability not have all channels active at once. Most users won't want multiple language channels active at the same time (neither for the audio or the text versions). Different mixes of active channels might be activated during a particular rendering of the presentation, depending on the requirements of the user. A Dutch-speaking blind person may not want video or text, but only the Dutch-language audio channel. An English speaking student of Dutch may want the Dutch audio and the English captions active (or vice versa), while a passenger on a crowded flight may want to see only the English captions and the video. Note that the choices do not need to be made solely by the user. If the playback environment realizes that streaming audio

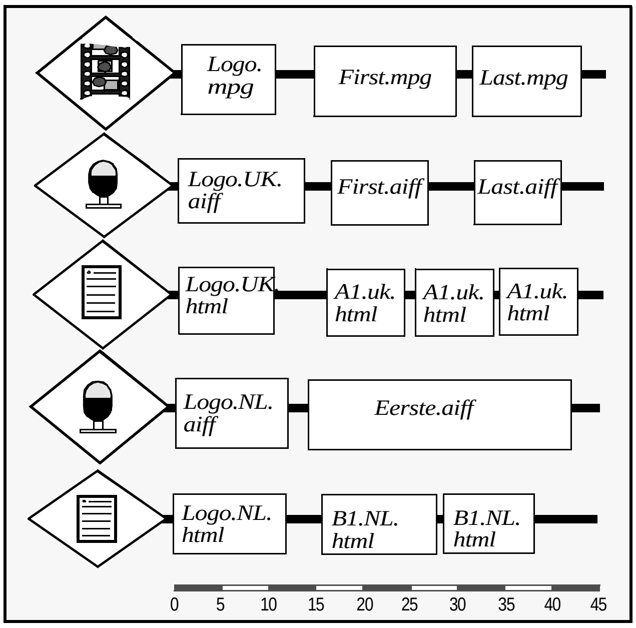


*Figure 2:* . Presentation fragment containing multiple *logical resource channel*s.

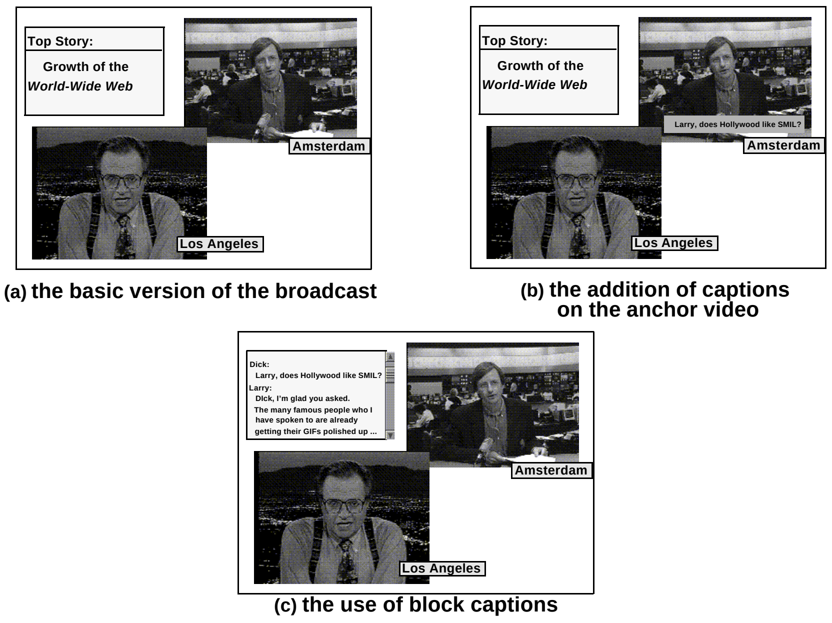
support is not available (or if the user has not made a micro-payment for the video), then it could itself choose to deactivate certain channels or to, say, substitute still images for the videos (assuming that such alternatives were specified by the author).

Figure 3 shows how the channel concept is used to support adaptivity. In Figure 3(a), we see a fragment from a network-delivered newscast. Figure 3(b) shows the same newscast with captions turned on for the main anchor (upper right) only. Figure 3(c) shows that line-by-line captions have been replaced by block captions in the upper left corner. The user can select which set of channels is to be active interactively during the presentation, or the selection can be based on that user's preferences profile.

From a CMIF perspective, the channel view is only one of the views that an author has on the relationships in a document. The main view is something that we call the hierarchy view; this is where the main temporal relationships are defined. The channels are simply management abstractions that can be widely applied to a variety of resource control needs.

## 2.3 W3C'S SMIL

SMIL, the Synchronized Multimedia Integration Language, is a W3C Recommendation for multimedia on the Web. During the development of the SMIL language [18], issue of selectability of content in a presentation received a great deal of attention. Early on, it was decided that a `SWITCH` construct would form the basic selection primitive in the encoding. A `SWITCH` allows a series of alternatives to be specified for a particular piece of content, one of which is selected by the runtime environment for presentation. An



Dick:
Larry, does Hollywood like SMIL?
Larry:
Dick, I'm glad you asked.
The many famous people who I have spoken to are already getting their GIFs polished up ...

**(a) the basic version of the broadcast**

**(b) the addition of captions on the anchor video**

**(c) the use of block captions**

*Figure 3:* . Three views of a newscast.

example of how a `SWITCH` might be used to control the alternatives that could accompany a piece of video in a presentation would be:

```
...
<par>
   <video src=”anchor.mpg” ... />
   <switch>
      <audio src=”dutch.aiff” ... />
      <audio src=”english.aiff” ... />
      <text src=”dutch.html” ... />
      <text src=”english.html” ... />
   </switch>
</par>
...
```

This fragment (which is pseudo-SMIL, for clarity) says that a video is played in parallel with one of: Dutch audio, English audio, Dutch text, or English text. SMIL does not specify the selection mechanism, only a way of specifying the alternatives.

The SMIL V1.0 recommendation currently supports both the notions of the `SWITCH` and a partial mechanism for controlling adaptive behavior called the *system test attribute*. The `SWITCH` provides a conventional branching structure that allows alternatives to be defined at authoring time. The system test attributes consist of a set of pre-defined (primarily system-related) attributes that describe dynamic aspects of the environment which can then be tested at run-time. For example:

```
<text src="cap.html" system-captions="true" .../>
```

will cause the object ‘cap.html’ to be rendered if *system-captions* evaluates to true.

The system test attribute mechanism is a significant extension over the SWITCH because of the way that it decouples the authoring and playback associations among a set of alternatives. Even so, it only partially meets the needs for adaptive control semantics because of the static nature of the attributes themselves (they are defined as part of the language, and can’t be extended easily by users).

## 3 ADAPTABLE APPROACHES

Adaptive presentations provide a basis for semantic processing of documents that are based on the alteration of individual components or components group within a portion of the presentation. Adaptable approaches provide a broader scope—they can cover an entire presentation—but typically require external processing to implementation the changes in presentation semantics. This section reviews three building-block technologies for adaptable presentations and then describes how they have been applied in the context of our Berlage environment.

### 3.1 THE SGML SUITE OF FORMATS AND THE SRM

Separation of structural and style information has long been commonplace for text, and can also be found in many hypertext models. In most hypermedia design models, including RMM [24] and HDM [12], the two types of information are designed during different phases of the design process. The primary implementation of the separation of structural and style information is found in the suite of SGML standards, which includes the suite of XML standards. The two SGML-related standards focussed on for this discussion are HyTime and DSSSL, each of which is described below. We also discuss the Standard Reference Model for intelligent hypermedia presentations, which defines a processing architecture for adaptable documents.

#### 3.1.1 *HyTime*

The ISO standard HyTime (Hypermedia/Time-based Structuring Language) specifies the representation of hypermedia documents in a presentation-independent format [19][11]. Because they are presentation-independent, HyTime documents must be processed into a presentation encoding for one particular renderer in order to be perceived by the user.

HyTime is an ISO standard for representing presentation-independent hypermedia data. HyTime is defined as a subset of Standard Generalized Markup Language (SGML) [21][13], which defines the structure of electronic documents in general. A related language that is also defined as an SGML subset is Extensible Markup Language (XML), which is a new format for providing a common framework for documents for different applications on the Web [4]. HyTime adds more complex structuring constructs and attaches hypermedia semantics to certain patterns of composites of this structure. The basic hypermedia semantics that HyTime represents include *hyperlinking*, which establishes descriptive relationships between document objects, and *scheduling*, which puts document objects in coordinate systems that can represent spatial and temporal structure.

HyTime and SGML are generally intended for encoding documents that are presentation-independent. They can apply to a wide variety of presentation situations but do not themselves represent particular presentations. HyTime and SGML documents typically must be processed into a different format appropriate for final presentation. HyTime and SGML are meta-languages. They encode not only individual documents but also the document sets to which they belong. A document set is defined by an SGML *document type definition (DTD)*. An individual document conforms to a particular DTD.

A DTD defines a specific syntax, in terms of SGML constructs, that its documents must follow. HyTime inherits from SGML the use of DTDs to define individual document sets.

### 3.1.2 *DSSSL*

The ISO standard DSSSL (Document Style Semantics and Specification Language) encodes the transformation between document storage and presentation [20]. DSSSL defines the transformation of SGML and HyTime documents into formats that present them. The use of DSSSL with HyTime was recently made easier with the release of the second edition of HyTime, which contains new facilities for use with DSSSL [19]. SMIL is defined as a subset of XML. Thus, DSSSL can encode transformations that output SMIL. HyTime documents are encoded to be adaptable, and DSSSL encodes how they are adapted.

DSSSL is a Scheme-like language that describes how an SGML document is transformed into another SGML document or into a non-SGML format. Because HyTime documents are SGML documents, any HyTime document can be transformed by DSSSL. A DSSSL program is typically called a *style sheet*. The separation of style from structure and content enforced with the distinction between DSSSL and SGML/ HyTime facilitates the creation of particular styles by the author that can be applied to documents of the same document set. Note that although the term style sheet is used, DSSSL can be used for more general, non-style transformations.

The design of typical DSSSL usage is shown in Figure 4. This diagram shows how an SGML document is processed with an accompanying style sheet by a DSSSL engine. The DSSSL engine determines the mapping encoded by the style sheet and generates the appropriate transformation of the source document into the presentation format. How this typical usage was implemented in the Berlage environment is described later in this paper. Also described later in this paper are the extensions to this typical usage required to implement dynamic generation of presentation.

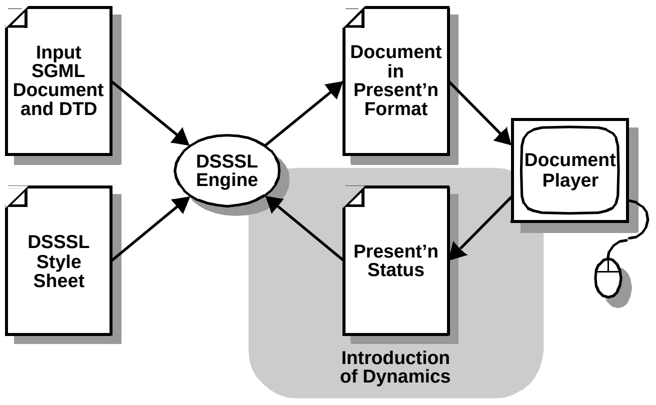


*Figure 4:* Typical Usage of DSSSL and the Introduction of Dynamics

### 3.1.3 *The Standard Reference Model*

Intelligent Multimedia Presentation Systems (IMPS) deal with the dynamic creation of multimedia presentations optimally geared towards the needs of a user. The Standard Reference Model (SRM) specifies the decomposition of an IMPS into well-defined layers [3]. Hypermedia built based on the SRM is thus very adaptive to the user, involving a rich set of adaptive semantics.

In broad terms, the created presentation should define what is presented to the user (the content), where it is presented (the spatial layout) and when it is presented (temporal layout). Given a collection of user goals and availability of resources these three aspects leave open an immense number of possible presentations. An IMPS is a reasoning system aimed at selecting the optimal one. The SRM can be used as the basic for discussing and comparing different systems that adapt to the user. It can also be used in guiding the development of such systems.

While SRM defines a general processing model for dynamic presentation generation and adaptation, existing formats and tools based on style sheets provide an existing infrastructure for performing similar functions. The difference is that style sheet-based environments often do not have dynamic adaptation, but static presentations are often generated that are adapted to circumstances that are known at generation time.

The SRM considers the decomposition of the dynamic creation of multimedia presentations into well defined layers. The SRM components are illustrated in Figure 5. The SRM divides dynamic presentation generation into two areas: *generation process* and *knowledge server*. The generation process performs the run-time generation of the presentation based on the user's interaction, the history of the presentation and information provided by the knowledge server. The knowledge server stores and provides long-term instructions and information that apply to multiple presentations at any point in their run. As such, the generation process is the active component of the SRM, and the knowledge server is the stable component.

## 3.2 THE SRM AND ITS IMPLEMENTION IN THE BERLAGE ENVIRONMENT

This section describes how the approaches described in this paper were applied to the design of the Berlage[a] hypermedia authoring and browsing environment [27]. Berlage incorporates the use of HyTime to represent hypermedia documents in a format independent of its presentation. It also incorporates the use of DSSSL to specify the different mapping of these documents to their final presentations. For the final presentation to the user, Berlage generates and hypermedia presentations encoded in SMIL. The Berlage environment consists of public domain tools to demonstrate how such environments can be readily implemented on a wide scale. The Berlage environment is designed in terms of the SRM and to specify layers of processing adaptation for the user during interaction with the presentation [25]. A diagram of the Berlage environment design is shown in Figure 6.

[a]The environment was named after H.P. Berlage, the leading 20th century architect of Amsterdam

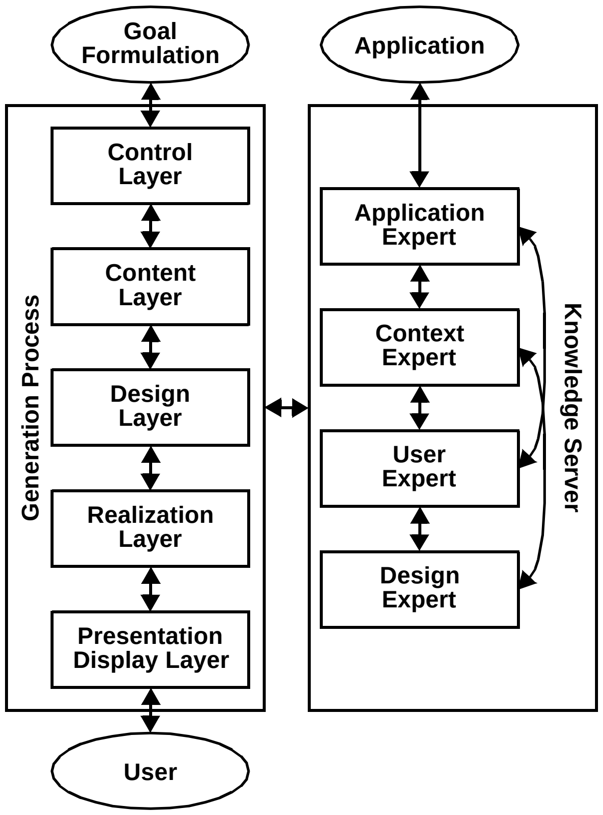


*Figure 5:* The SRM and its Components

Before the incorporation of the SRM, the Berlage environment created only static presentations. With the SRM incorporation performed for this paper, the Berlage environment can adapt the presentation during the presentation itself. The dynamic generation of DSSSL code required by the SRM implementation necessitated the substantial additions and modifications to the Berlage environment architecture, as shown in Figure 6. The primary new component added is the http server. The server enables the Berlage environment to keep track of the user's interaction with the presentation and with each interaction generate a new presentation state in the form of DSSSL code. The environment then calls the Jade DSSSL engine to again process the document with the main DSSSL style sheet, which includes by reference the newly generated presentation state code. This results in the creation of new SMIL code, which is then passed back to the SMIL player client by the http server.

To enable the Berlage http server to keep track of the user interaction, the destination of all navigational hyperlinks in the SMIL code specifies with URL code that http server. Further, these URL addresses are in the format used by HTML forms. Fields and values for those fields are set in the URL to convey to the server what the status of the presentation was and what choice among those available the user selected. The server can then update its internal data store to properly represent the presentation's state. Next it generates the DSSSL presentation state code for inclusion in the next Jade processing of the document and its presentation. This generated DSSSL code includes data that enables the style sheet to define SMIL hyperlink destination URLs so that the server will recognize them when they are activate and can respond appropriately.

The SRM's division between the generation process and the knowledge server is maintained in the Berlage environment. The data fed to the layers of the generation process, except for the presentation display layer, are represented as presentation status code. In the Berlage environment SRM extension, this code is DSSSL code that is included in the main style sheet for the document and its current presentation. The instructions for how each layer handles the data in this presentation state code are in the static main style sheet DSSSL code.

At least some of the processing of the experts can also represented by DSSSL code in the main style sheet. The code would typically reference the storage document HyTime structure code, while the generation process code references the presentation status more. DSSSL as a language has been adequate for this. More complex decisions processes may require software outside of DSSSL to complement what is provided in the Berlage environment.

### 3.2.1 *Control Layer*

This layer selects a goal from a set of goals which have still to be met. It influences the message to be expressed by a presentation, but does not make any decisions directly affecting the instantiation of parts of a particular presentation.

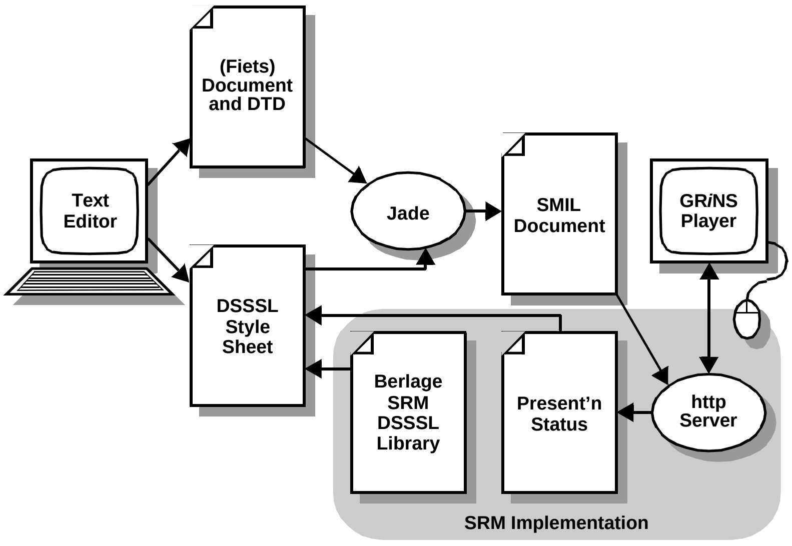


*Figure 6:* Berlage Environment Design with SRM Implementation

For example, Figure 7 shows a fragment of a presentation called *Fiets*, which gives an adaptable tour of Amsterdam. In the view shown, the user is given a collection of Amsterdam buildings about which more detailed information can be accessed. A thumbnail image of the front of each building is shown, and each image is the starting point for a hyperlink leading to more information about that building. It is a goal of the application that the user visit each of the buildings. The user is helped in this task by the changing appearance of this portion of the Fiets presentation. After the user has visited a building and returns to this screen, the visited building, and all previously visited buildings, are shown with a more lightly shaded border. The user can then see that the buildings with darker borders are ones that still need to be visited.

### 3.2.2 *Content Layer*

In the SRM, the content layer takes a single goal passed on by the control layer and refines it to a list of subgoals. These are in turn transformed into a set of communicative acts and relations among these acts. This is carried out by four separate, but communicating processes—*goal refinement*, *content selection*, *media allocation* and *ordering*. Goal refinement is a task which has to be carried out before any parts of a presentation are created, so it is not discussed further here.

The content selection process communicates with the application expert, and decides on the semantic content of the presentation. In the Fiets presentation, when the user has selected a building the content of the resulting SMIL display must be determined. This display shows detailed information about a building, thus such content may include history of the building such as when it was built and whom its residents were. This

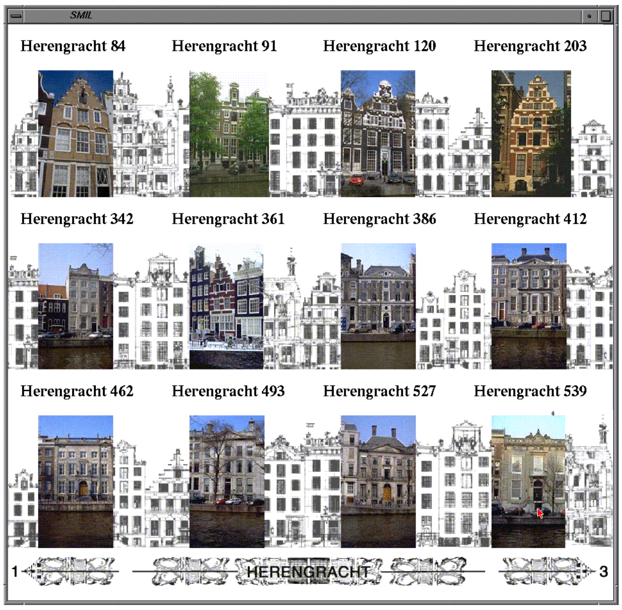


*Figure 7:* GR*i*NS Showing a Fiets Presentation

display could also convey more detailed information on how the building appears on the outside and on the inside.

When the http server receives a URL request for detailed information on a particular building, it conveys in generated DSSSL code what the appropriate semantic content is to display for that building. This would typically be represented by the setting of variables similar to those used for the control layer. In the Fiets example being discussed, the main style sheet could access these settings conveying what semantic content should be in the next display. For each unit of semantic content, the main style sheet could have a set of static instructions on how to display it.

The media allocation component assigns a particular medium to a communicative act, producing a media communicative act. In the Berlage environment, this corresponds to the selection of particular media objects to display for a given semantic content. In the Fiets example, the semantic content of detailed information on a buildings exterior could correspond with particular images of its gable ornamentation and entranceway.

The instructional code for determining this media allocation would be contained in the main style sheet, not in the generated state DSSSL code. The main style sheet when given a unit of content to convey has the instructions for determining the media allocation for that content. In Fiets, such instructions often refer to the HyTime-defined structure of the stored document to determining this media allocation. For example, HyTime structures in Fiets associate images of gable ornamentation and entranceways with individual buildings. This HyTime code would be accessed by Jade as instructed by the main DSSSL style sheet to determine these image files for a particular building.

### 3.2.3 *Design Layer*

The design layer allows the processes of media selection and layout design to carry on in parallel, imposing no ordering requirements for which of these should be finalized first. The design layer is split into two communicating processes: *media design* and *layout design*. The media design component makes decisions on the "look and feel" of the presentation. It can communicate with the design expert and make decisions on styles. Style information can include as color, font, background wallpaper images, and other aspects of the presentation that do not affect the actual media content or its positioning in space and time in the presentation. In the Berlage environment, the design expert can be embodied, at least in part, by code within the main style sheet.

In the Fiets example, one screen display has thumbnail images for multiple buildings. The number of thumbnail images to be displayed on a single screen is a design layer spatial layout decision. If there are too many buildings to select from, the thumbnails are distributed among multiple screen displays. Access to each of these displays is provided with hyperlinks. The number of thumbnails that can fit on a single display is determined in the SRM by the design expert. In the Berlage environment, it is code in the main style sheet that states this number, and this code corresponds to the SRM design expert.

### 3.2.4 *Realization Layer*

In this layer the final decision is taken on the media items to be played in the presentation and their corresponding spatial and temporal layout. Again, the layer is split into two communicating processes, paralleling those in the design layer—media realization and layout realization. The media realization component ensures that appropriate media items are chosen. These may already exist, or may be generated specifically for the task.

The layout realization component calculates the final temporal and spatial layout of the presentation based on the constraints specified in the design layer. The duration in time of an atomic media object's presentation may be derived from the content for continuous media or from the storage structure. The duration of a composite portions of the presentation can be calculated on the basis of its components along with any other temporal constraints specified in the design layer. Where links define possible paths among separate multimedia presentations, there is also a duration perceived by the user when the link is followed.

In the Fiets example, the exact position of each building thumbnail would be determined. This positioning will give the building images an even distribution in the screen space. The exact font size and positioning of the associated text and hyperlink hotspots would also be determined here. The design layer determines how may building thumbnails to show at once, while the realization layer determined where on the screen to show them.

#### 3.2.5 *Presentation Display Layer*

In the Berlage Environment the presentation display layer is embodied by the generation of the SMIL code, which is played by GR*i*NS, that displays the presentation to the user.

## 4 WHERE SHOULD PRESENTATION ADAPTATION AND ADAPTABILITY SEMANTICS BE SUPPORTED?

### 4.1 REFLECTIONS ON ADAPTIVE HYPERMEDIA

The primary difference between adaptive and adaptable presentations is the degree to which the adaptation process occurs autonomously. In adaptive presentations, the presentation author must account for the transformation of the presentation at author-time. As mentioned in section 2, this can happen either based on executable program code or via a declarative means.

Perhaps the most problematic aspect of using script-based control for adaptivity is that most document authors are not programmers. Even if they were, the user-centered nature of the adaptive process makes the task of integrating system- and user-needs with a single code fragment sufficiently complex that such an approach would serve as a disincentive to creating generalized alternative presentations. In the context of a full presentation, it is often difficult to define the impact of all of the various user dependencies 'up front' at author-time: if a user wanted to tailor only a special portion of a presentation or if they wanted to make unusual choices (supporting multiple languages simultaneously), this is typically difficult to predict in advance.

In general, declarative forms of adaptation control are much more flexible for the author and the runtime environment. In essence, a declarative approach provides a catalogue of needs, allowing the user and the environment to indicate which needs are relevant for any one presentation. With a declarative approach, the question is often not ‘how do I support adaptivity’, but rather: ‘at what level in the processing hierarchy should such support take place.’ This relates to the granularity of the storage model as well as the control model of the presentation.

In section 2, we discussed the use of a fully-generated presentation via a database interface, as well as simplifications on this model. In our experience, the use of database back-ends for total processing often are more appropriate for adaptable presentations than for adaptive. Inherent in the processing of adaptive presentations is the need to coordinate several concurrent streams of information, each with its own logical focus. If we add to this the inherently distributed nature of the Web infrastructure and the unpredictable semantic behavior of a hypermedia application (where rescheduling or regenerating the next presentation fragment is only a mouse-click away!), then we have yet to find a suitable information model that can effectively replace the insights of a presentation author.

Where storage-based adaptivity is appropriate is in fetching a version of a media object for which there may be many (semantic) encodings. This is the approach taken in part by Transparent Content Negotiation and similar systems. While TCN represents an improvement over the simple line-item substitution in, say, HTML, it typically hides the entire substitution process. Users have little control, since users don’t get to see and evaluate the alternatives. Also, since each object is developed independently, defining common user-level grouping abstractions across these independent objects is a difficult problem.

Our experience with CMIF has led us to support the notion of author-based logical grouping of semantic alternatives for adaptive hypermedia. Constructs like CMIF’s Channels provide a user-centered alternative in which a user (or user’s agent) can evaluate the alternative available and select the most appropriate encoding. This has proved useful in the areas of multi-lingual presentations (where the exact combination of languages used in a single presentation can vary: some users want natural audio and customized subtitles, while others want dubbed-audio and no captions—but often, the exact choice depends on the nature of the object being viewed rather than static preferences). It has also proven useful for serving the needs to the accessibility community, where standard documents can be adapted using standard tools.

We are pleased that SMIL provides a relatively high degree of support for adaptability through the `SWITCH` and the system test attributes. While this is a major step forward, it is still incomplete for general presentations. The `SWITCH` has two problems. First, it restricts the resolution of a `SWITCH` to a single alternative. (If you want Dutch audio *and* Dutch text, you need to specify a compound `SWITCH` statement, but in so doing, you always get the compound result.) More restrictively, it requires the author to explicitly state all of the possible combinations of input streams at author time. If the user wanted Dutch audio and English text, this possibility must have been considered at authoring time. If we compare the SWITCH construct with CMIF’s Channels, it is true that, logically, the alternatives indicated by the channel construct could be represented as a set of `SWITCH` statements, although the resulting size of the `SWITCH`’s composite structure would become explosive.

We feel that the SWITCH is of most use for syntactic alternatives. The test attribute option is a richer alternative, although SMIL missed an opportunity to focus on the semantic needs to the user rather than that of the encoding. Use of a Channel-like mechanism would significantly simplify the specification of user-centered alternative. The author could specify all of the individual components of a presentation at author time and then organize them in terms of their logical channel threads. The user (or user's agent) can then select which sets are active at runtime. An 'initial state' attribute can be used to define the default behavior.

## 4.2 REFLECTIONS ON ADAPTABLE HYPERMEDIA

The options available to supporting adaptable hypermedia can best be understood in terms of the SRM.

The role of the control layer in supporting adaptable presentations is typically restricted to processing documents (or document fragments) based on the current state of a presentation. For example, when the user selects a building in the Fiets example of section 4, the SMIL browser client sends an http request to the Berlage http server. This signals to the server that a particular building was selected. The server then generates presentation state DSSSL code that when included in the main style sheet for processing causes SMIL code to be generated that displays detailed information on that building. In this and all future presentation state DSSSL code generated, a boolean variable is set for that building indicating that it has been visited. The main style sheet has instructions that access this code in determining what shade border to use for each building in generating the SMIL code representing the next step in the presentation. The purpose of control operation adaptability is often to reflect the navigational state in terms of the document structure rather than altering the semantics of the presentation.

At the content layer, the issues is often not one of data granularity (as was the case with adaptive substitution) but more one of ranging and ordering choices. In the Fiets example (and in Berlage, in general) the result of the ordering process is a (not necessarily linear) ordering of the media communicative acts. Navigational hyperlinks, timing relations and spatial layout are all different ways of expressing ordering relations among media items in the final SMIL presentation code. When a communicative act is to be expressed using a number of media items these methods can be traded-off against one another. For example, rather than display many items together, links can be made among smaller groups of the items, or the sequential playing of items may be a suitable alternative for laying out items next to each other at the same time [28]. The choice among these alternatives in the Fiets application is described in other work [26]. These different ways of "ordering" the material need to be captured within the content layer, but may not be finalized until the design or realization layer. The content layer is thus restricted to specifying a structure of links, and leave the decisions on temporal and spatial layout to other layers. In the Berlage environment, specifying this link structure for a given state in the presentation means determining what to show on the display being generated and what information should instead be made accessible by links from this generated display.

While the design and realization layers have little direct influence on adaptability, a much more significant role is reserved for the presentation display layer. In nearly all previous systems, there has been a very tight coupling of presentation environment to the abstract model of the presentation domain. While HyTime and SGML provide support for decoupling the document for the presentation, the reality of most time-based (hypermedia) systems is that the development of the abstract model is often constrained by the presumed system interface. In this regard, the increasing number of options available to content developers for display of hypermedia information may provide a catalyst for developing cross-environment presentations. For example, a SMIL presentation generated for the Web domain may also be reused as part of a broadcast TV commercial, or a direct broadcast news program may be augmented with links that are only appropriate for viewing in a later on-demand mode.

In this latter case, the meaning of the presentation relative to its distribution environment, more so than its encoding, will form a fundamental requirement for including semantic-based adaptive processing of the presentation that will transcend the reliance on representation-based manipulations which form the basis of current adaptability work.

## 5 SUMMARY

This paper described various hypermedia approaches and how each handles the distinction between the semantics of adaptive and adaptable hypermedia. These approaches included primarily SMIL, CMIF, the SGML suite of standards and the SRM. Experience learned from applying these approaches to developing the Berlage environment and Fiets application was described. Conclusions were drawn from these approaches regarding when it is best to apply adaptive semantics and when it is best to apply adaptable semantics in designing hypermedia. The impact of this choice on the database performance was also described.

## ACKNOWLEDGMENTS

The GR*i*NS environment was implemented by Sjoerd Mullender, Jack Jansen and Guido van Rossum. The Fiets artwork and graphic design was done by Maja Kuzmanovic. The development of GR*i*NS was funded in part by the European Union ESPRIT Chameleon project. The images used in this paper come from the Amsterdam Heritage Site [1].

## REFERENCES

1. City of Amsterdam Municipal Department for Preservation and Restoration of Historic Buildings and Sites. *Amsterdam Heritage*. http://www.amsterdam.nl/bmz/adam/adam_e.html.
2. Böhm, K., Aberer, K., and Klas, W. Building a Hybrid Database Application for Structured Documents, *Multimedia Tools and Applications*, 1997.
3. Bordegoni, M., Faconti, G., Feiner S., Maybury, M.T., Rist, T., Ruggieri, S., Trahanias, P. and Wilson, M. A Standard Reference Model for Intelligent Multimedia Presentation Systems, *Computer Standards and Interfaces* 18(6,7) December 1997, North Holland, pp. 477-496.
4. Bray, T., Paoli, J. and Sperberg-McQueen, C.M., *Extensible Markup Language (XML)*. W3C Recommendation, January 1998. http://www.w3.org/TR/REC-xml.html.
5. Buford, J.F., Rutledge, L., Rutledge, J.L., and Keskin, C. HyOctane: A HyTime Engine for an MMIS, *Multimedia Systems*, vol. 1, no. 4, February 1994.
6. Buford, J.F., Rutledge, L. and Rutledge, J.L. Towards Automatic Generation of HyTime Applications, in Proc. *Eurographics 94*, June 1994.

7. Bulterman, D.C.A. Models, Media and Motion: Using the Web to Support Multimedia Documents, in Proc. *Multimedia Modeling 97*, November 1997.

8. Bulterman, D.C.A., Hardman, L., Jansen, J. Mullender, K.S. and Rutledge, L. GRiNS: A GRaphical INterface for Creating and Playing SMIL Documents, in Proc. *Seventh International World Wide Web Conference (WWW7)*, April 1998.

9. Bulterman, D.C.A. User-Centered Abstractions for Adaptive Hypermedia Presentations, in Proc. *ACM Multimedia 98*, September 1998.

10. Clark, J. *Jade — James' DSSSL Engine*. http://www.jclark.com/jade/.

11. DeRose, S. and Durand, D. *Making Hypermedia Work: A User's Guide to HyTime*. Kluwer Press, Boston. 1994.

12. Garzotto, F., Mainetti, L. and Paolini. P. Hypermedia Design, Analysis, and Evaluation Issues, *Communications of the ACM*, 38(8):74 86, August 1995.

13. Goldfarb, C. *The SGML Handbook*. Oxford University Press. 1991.

14. Graf, W., Intelligent Multimedia Layout: A Reference Architecture, *Computer Standards and Interfaces*, 18 (1997).

15. Hardman, L., Bulterman, D.C.A. and van Rossum, G. The Amsterdam Hypermedia Model: Applying Time and Context to the Dexter Model, *Communications of the ACM*, vol. 37, no. 2, February 1994.

16. Hardman, L., Worring, M., and Bulterman, D.C.A., Integrating the Amsterdam Hypermedia Model with the Standard Reference Model for Intelligent Multimedia Systems, *Computer Standards and Interfaces*, 18 (1997).

17. Hotlman, K. and Mutz, A. *Transparent Content Negotiation in HTTP*, IETF RFC 2295. ftp://ftp.isi.edu/in-notes/rfc2295.txt

18. Hoschka, P. (ed.). *Synchronized Multimedia Integration Language*. W3C Recommendation, June 1998. http://www.w3.org/TR/REC-smil/.

19. International Standards Organization. *Hypermedia/Time-based Structuring Language (HyTime)*. Second Edition. ISO/IEC IS 10744:1997, 1997.

20. International Standards Organization. *Document Style Semantics and Specification Language (DSSSL)*. ISO/IEC IS 10179:1996, 1996.

21. International Standards Organization. *Standard Generalized Markup Language (SGML)*. ISO/IEC IS 8879:1985, 1985.

22. MONET: *Extending databases for multimedia*. http://www.cwi.nl/~monet/modprg.html

23. Özsu, M.T., Iglinski, P., Szafron, D., El-Medani, S. and Schöne, M. An Object-Oriented SGML/HyTime Compliant Multimedia Database Management System, in Proc. *ACM Multimedia 97*, November 1997.

24. Isakowitz, T., Stohr, E.A., and Balasubramanian, P. RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8):34 44, August 1995.

25. Rutledge, L., Hardman, L., van Ossenbruggen, J. and Bulterman, D.C.A. Implementing Adaptability in the Standard Reference Model, in Proc. *Multimedia Modeling 98*, October 1998.

26. Rutledge, L., Hardman, L., van Ossenbruggen, J. and Bulterman, D.C.A. Structural Distinctions Between Hypermedia Storage and Presentation, in Proc. *ACM Multimedia 98*, September 1998.

27. Rutledge, L., van Ossenbruggen, J., Hardman, L. and Bulterman, D.C.A. Practical Application of Existing Hypermedia Standards and Tools, in Proc. *Digital Libraries 98*, June 1998.

28. Weitzman, L. and Wittenburg, K., Grammar-Based Articulation for Multimedia Document Design, *Multimedia Systems*, 4: 99-111, 1996.

29. W3C: HTML 4.0: W3C's Next Version of HTML. URL: http://www.w3.org/MarkUp/Cougar/