Mix'n'Match: Exchangeable Modules of Hypermedia Style

Lloyd Rutledge, Lynda Hardman, Jacco van Ossenbruggen and Dick C.A. Bulterman
CWI (Centrum voor Wiskunde en Informatica)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Tel: +31 20 592 4127
E-mail: {Lloyd.Rutledge, Lynda.Hardman, Jacco.van.Ossenbruggen, Dick.Bulterman}@cwi.nl

ABSTRACT
Making hypermedia adaptable for multiple forms of presentation involves enabling multiple distinct specifications for how a given collection of hypermedia can have its presentation generated. The Standard Reference Model for Intelligent Multimedia Presentation Systems describes how the generation of hypermedia presentation can be divided into distinct but cooperating layers. Earlier work has described how specifications for generating presentations can be divided into distinct modules of code corresponding to these layers. This paper explores how the modules for each layer of a presentation specification can be exchanged for another module encoded for that layer and result in the whole specification remaining well functioning. This capability would facilitate specifying presentation generation by allowing for the use of pre-programmed modules, enabling the author to focus on particular aspects of the presentation generation process. An example implementation of these concepts that uses current and developing Web standards is presented to illustrate how wide-spread modularized presentation generation might be realized in the near future.

KEYWORDS: Adaptable hypermedia, presentation specification, IMMPSs

INTRODUCTION

Enabling stored hypermedia to be adapted to a wide variety of presentation circumstances enhances its reusability and reduces its need for future editing. Presentation circumstances that necessitate adaptation include user preferences and abilities, system resources and constraints, and what the presentation is intended to convey to the user. This paper uses a distinction between adaptive and adaptable hypermedia that is described in earlier work [5]. Here, adaptive hypermedia accounts for all anticipated variations and directly encodes the response to all possibilities, making it directly presentable but complicating its adaptation to unforeseen presentation circumstances. Adaptable hypermedia, on the other hand, does not specify any presentation but instead encodes the information needed to generate appropriate presentations from it. Adaptable hypermedia is more readily transformed to a wider variety of presentations than adaptive hypermedia, but requires separate specifications for how a presentation is to be generated from it. This paper focuses on adaptable hypermedia and the specifications it requires for generating presentations.

This paper refers to the specification of presentation generation as *style*. Here, style refers to more than details of the visual appearance of the final presentation. It refers to all the choices that are made in determining how a collection of hypermedia is to be presented. That which style determines includes visual appearance, screen layout, media selection, timeline and navigational interface.

Multiple styles can be written for a single collection of hypermedia, each resulting in a different presentation of the same material. The user can choose the most favorable style for application to the hypermedia to be presented. Each style can be tailored to the needs and preferences of the user or to the characteristics of the current presentation environment, or both. Styles can be written that take into account presentation circumstances that the original hypermedia author could not have anticipated.

Rather than treating a style as an atomic unit, this paper explores the division of styles into exchangeable modules. This gives the user more flexibility in determining a style with which to be presented a hypermedia document. Rather than selecting one entirely prefabricated style, the user generates his or her own presentation with this paper's *Mix'n'Match* scheme. With this scheme, the user can select an appropriate substyle module for each layer and then stack these selections to create a new style better suited for his or her needs. Thus, the number of different presentation specifications a user can apply to a document is not limited to the number of styles but to the number of different combinations of style modules that can applied to that document. Further-

more, as standards and products for presentation generation evolve and develop, an understanding of how to modularize their functions will help in determining how best to define them.

The Fiets hypermedia application is used in this paper to illustrate the issues described. Fiets (the Dutch word for "bicycle", pronounced *feets*) is an hypermedia tour through the architecture and history of the buildings in Amsterdam, The Netherlands [21]. The Berlage environment design is used to demonstrate how the layering of exchangeable style modules can be implemented using existing and developing Web standards and publicly available tools [23]. The Fiets application is implemented in the Berlage environment.

This paper uses the Standard Reference Model for Intelligent Multimedia Presentation Systems (SRM-IMMPSs) [2] as the basis for dividing hypermedia style into layers. The SRM-IMMPSs defines layers of the generation of multimedia presentations. It describes what each layer does and how the layers cooperate with each other to dynamically generate a presentation tailored for the current user and environment. The application of the SRM-IMMPSs to the Berlage environment is described in earlier work [22].

This paper extends earlier work on Fiets, Berlage and the SRM-IMMPSs by describing how the Mix'n'Match scheme described herein can be applied to them. The goal of this application is to introduce the exchangeability of style modules to Berlage as a vehicle for exploring the exchangeability of style modules for hypermedia in general. Fiets is used here as an illustrative example. It has been modified for this paper to demonstrate the Mix'n'Match's use of distinct modules of style code and the effect of exchanging. This paper also describes the changes made to the Berlage environment to provide this exchangeability. The layering and components of the SRM-IMMPSs is the basis for the modularization presented in this paper. This paper describes how the use of the SRM-IMMPSs applies to the implementation of Mix'n'Match. It also describes what the effect of this implementation of exchangeability was on the application of the SRM-IMMPSs to Berlage. First some background material is provided, giving a broad overview of the SRM-IMMPSs and the Berlage environment. Then the implementation of Mix'n'Match into each component of the Berlage environment is described.

BACKGROUND

Standard Reference Model for Intelligent Multimedia Presentation Systems (SRM-IMMPSs)

In broad terms, a hypermedia 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 wide variety of possible presentations. An Intelligent Multimedia Presentation System (IMMPS) is a reasoning system aimed at selecting the optimal one. The SRM-IMMPSs is a software architecture designed to be used as the basis for discussing and comparing different systems that adapt to the user. It can also be used in guiding the development of such systems.

The SRM-IMMPSs considers the decomposition of the dynamic creation of multimedia presentations into well defined components. These components are illustrated in Figure 1. The SRM-IMMPSs 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 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. It also keeps track of the history of the presentation. With this division of function, the generation process is the active component of the SRM-IMMPSs, and the knowledge server is the stable component. The generation process is divided into layers, each of which controls one aspect of dynamically generating the presentation. The knowledge server is divided into experts, each of which stores information about one aspect of what the generation needs to take into account.

The division into layers and experts that the SRM-IMMPSs provides serves as the basis for this paper's division of hypermedia style into modules. In the following subsections the components of the SRM-IMMPSs are discussed one at a time, as is the impact of each component on the modularization of style. Typically, each component of the SRM-IMMPSs corresponds to a distinct type of style module. This paper refers to the primary introductory publication of the SRM-IMMPSs for readers who want more detailed information on it [2].

The Berlage Environment

The Berlage environment architecture was designed to demonstrate the implementability of adaptable hypermedia with existing standards and tools [23]. Its design was extended to match that of the SRM-IMMPSs [22]. The current design of the Berlage architecture is illustrated in Figure 2. Its components are each briefly described below. Earlier work describes its components in more detail [23][22].

HyTime is used to encode the presentation-independent hypermedia semantics of the stored documents [17][10]. HyTime is a syntactic subset of *SGML* [19][11], extending the semantics SGML encodes into hypermedia. HyTime syntax is defined as an *SGML architecture* with a *meta-DTD* [17]. The semantics encoded by HyTime constructs in a document can be queried for using the *property sets* defined in the HyTime standard. HyTime was used in Fiets to represent such relatively common hypermedia semantics as a year in history.

An SGML architecture defines a broad set of SGML and HyTime documents that share semantics within a particular conceptual domain. Its syntax is defined with a meta-DTD. Access to the semantics represented by its syntax is defined

with property sets. SGML architectures can inherit from one or more other architectures, and inherit their property sets. An SGML architecture called Berlage is defined for the Berlage environment [24]. It includes such common hypermedia semantics as dimensions in time or pixels of a media object.

A *DTD* more narrowly defines the syntax of a document set. Property sets can be defined for DTDs as with architectures. This syntax of the Fiets document collection about Amsterdam is defined as a DTD. An individual document can be validated in terms of all levels of syntax described above with the tool *SP* [7].

DSSSL is a lisp dialect encoding the transformation of SGML and HyTime documents into other SGML documents that are typically directly processed for presentation [18]. DSSSL programs are called *style sheets*, providing the basis for the term "style" as used in this paper. DSSSL provides inclusion mechanisms that enable the division of DSSSL code into libraries shown in Figure 2. DSSSL can query against HyTime-defined properties. Some of the *DSSSL libraries* in Fiets define the functions that process these properties, a technique which is described in the initial work on the Berlage environment [23]. *Jade* is a publicly available DSSSL engine [6].

SMIL is an XML-compliant, HTML-like W3C recommendation for hypermedia on the Web [16]. SMIL is easily processed as output by DSSSL because it is encoded as *XML*, which is a subset of SGML. *XP* is an XML parser which is used here to validate SMIL code [8]. Berlage provides DSSSL functions to facilitate the generation of SMIL output. *GRiNS* is a publicly available player for SMIL presentations [4][9].

Part of the incorporation of the SRM-IMMPSs into Berlage involved introducing dynamics [22]. This is done with an http server that outputs DSSSL encoding the user interaction history of the presentation. This DSSSL code representing the *presentation status* is then incorporated into the style sheet for processing the next step of the presentation.

The style modules from the Mix'n'Match scheme would be represented in the Berlage environment as separate files included by reference in the main Fiets DSSSL style sheet. These DSSSL style module files can be stored locally or accessed from anywhere on the Web with URLs and Jade's processing of them. This enables the distribution of style as well as content.

GOAL FORMULATION

The collection of *goals* of an SRM-IMMPSs-modeled presentation it what determines when a presentation to the user is complete. Each presentation for a user has a purpose, a body of information that is to be conveyed to the user so that the user understands it. The processing these goals is the most important part of how the SRM-IMMPSs works for this processing is the basis of how the presentation of the document progresses to its completion. The goal formulation component of the SRM-IMMPSs handles the initial interac-

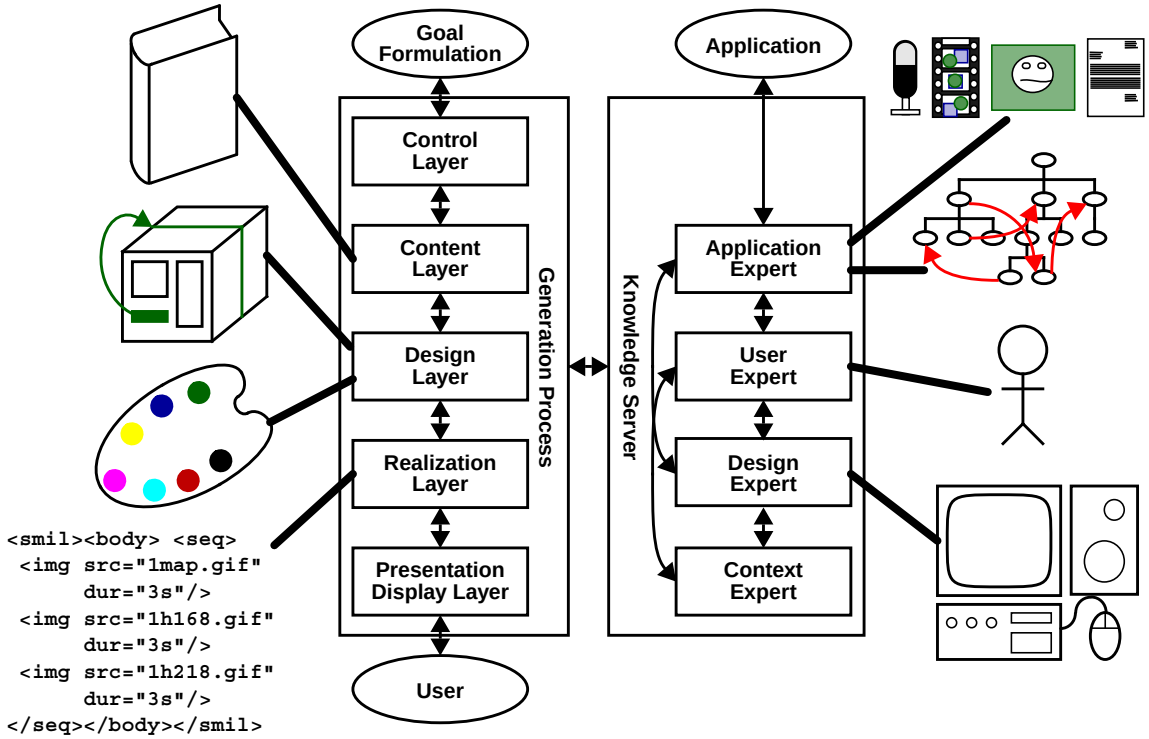

Figure 1: The SRM-IMMPSs [2] and its Use in Mix'n'Match

tion with the user that starts the presentation and establishes its goals. These goals are broken up into subgoals which are then passed one at a time to the control layer, which makes sure that each is met.

The goal formulation activity in the Berlage architecture is the establishment for a given presentation of the main style sheet to use for it, with the inclusions for each style module and the required DSSSL libraries. Goal formulation activity would also include loading the style sheet for processing by Jade for the first step in the presentation.

CONTROL LAYER

The control layer activity in the Berlage architecture consists of determining what subgoals to meet next through Jade's processing of the style sheet during subsequent steps, as delimited by user interaction, of the presentation. This behavior is encoded in the Fiets style sheet, as shown in Figure 2, though its inclusion of the exchangeable style modules. The Fiets style sheet also contains DSSSL code that applies to any instance of each exchangeable module included with it for processing. The Fiets style sheet works for documents of a specific topic domain: the architectural history of Amsterdam.

All the DSSSL-encoded modules used with the example described in this paper work specifically for this Fiets style sheet and have not been made not apply to other topic domains. It is up to future research to explore how broad the topic domain can be that a primary style sheet and the modules included with it will work appropriately for.

CONTENT LAYER

In a paper-based society, a person reads a document to understand a particular body of knowledge. He or she can expect to be able to communicate about this topic with other people who have read the same paper document. The same common understanding should be expected for two people who have been presented the same hypermedia document, no matter how much the document may have been adapted differently for each user The content layer processes the set of goals that must be met in order for a particular understanding to have been communicated by the presentation.

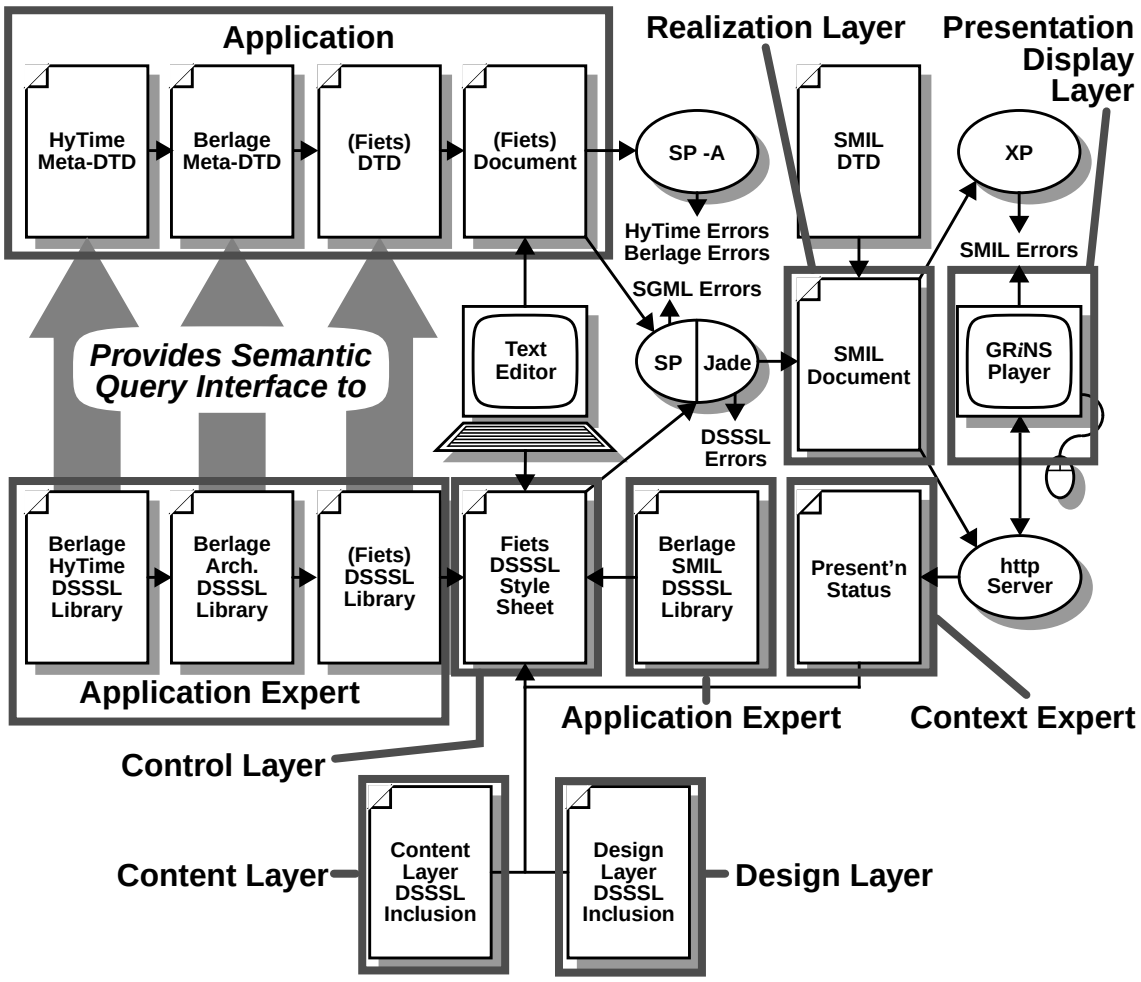

Figure 2: Berlage Environment Architecture [23][22] and its Use for Mix'n'Match

A DSSSL-encoded content layer module could exist on the Web and act as what most people would perceive as the "document". Including this module in a style sheet is what determines the information conveyed by the presentation. The information itself is stored as SGML and HyTime code, but may not necessarily coincide with the user's perception of the "document" because this store of information may be immense, with only a small portion of it selected by style sheet processing for presentation to the user as the "document" he or she requested.

Figure 3 shows a display from the Fiets "panorama" style presentation, in which multiple building images are displayed at once, distributed across the screen. The content layer module for this presentation is the "by address" module. The "by address" content layer module states that the presentation's overall goal is to view all of the houses along the Herengracht in a manner that conveys how these houses change as one proceeds south along the canal. A separate "by year" content layer module may generate a presentation conveying a sense of how the appearance of these houses tends to vary depending on when they were first built rather than where on the canal they are located. These two types of presentation are described further in the initial work on Fiets [21]. The key difference between these presentations is that each presents the same content in different orders. Determining presentation order is one of the key tasks of the content layer, and thus of a style module that corresponds with it.

With the processing of Fiets by the Berlage architecture, content layer style modules "by address" and "by year" are files of DSSSL code that are included in the main DSSSL style sheet used for starting and running a presentation. The initial work on the Berlage environment architecture presents sample DSSSL code that determines the order in which Fiets

Figure 3: A Fiets "Panorama" Style Presentation [21]

objects are presented [23]. This code queries the HyTime-defined structure that defines the street addresses of selected buildings to generate a list of these buildings in order of their street address. Thus, this code defines the primary processing of the "by address" module. The data structure that results from this processing is what is passed to the design layer for further processing.

Figure 4 shows the DSSSL code from a Fiets content layer style module which generates "by address" presentations. This code simply assigns values to some constants used in other modules that determine the "by address" sorting of document information in the presentation generated for the user. This code states that the Fiets-define property to sort on is "street-nbr". It also states that the HyTime FCS (Finite Coordinate Space) construct from which to get this sort information is the one defining the "STREETFCS" axis along which buildings are positioned. In the "by year" content layer style module, the DSSSL code assigns these same variables to "year" and "YEARFCS".

The Amsterdam Hypermedia Model (AHM) [14], which extends the Dexter Hypertext Model [13] into hypermedia, has been applied to the SRM-IMMPSs [15]. This work describes what components of the AHM describe the activities of the components of the SRM-IMMPSs. AHM and Dexter are shown here to contribute to the SRM-IMMPSs by providing a more detailed model for hypertext and hypermedia documents. AHM and Dexter also provide constructs for guiding how components of the documents they model are to be presented to the user. These AHM and Dexter concepts can be applied to the creation of style modules. They apply content layer by defining attributes for media objects that represent their semantic content [15]. These attributes are then used as the basis for selecting the content that is appropriate for the presentation.

DESIGN LAYER

The design layer of the SRM-IMMPSs determines what means the final presentation uses to meet the goals established in the content layer. This layer makes decisions on the "look and feel" of the presentation, including what media objects should be selected and how they should be presented. The design layer also determines the spatial-temporal and navigational layout of the final presentation.

The Fiets application is used in earlier work to illustrate how there may be many different means to convey the same aspects of a stored hypermedia document [21]. The different style combinations of storage and presentation structure that result in Fiets are shown in Figure 5. Each of the nine presentation styles that result are described in this earlier work.

```
(define ContentSortProp street-nbr )
(define ContentFCS      "STREETFCS")
```

Figure 4: Example of Fiets "by address" Content Layer Style Module DSSSL Code

Then, these presentations were generated by nine different style specifications. This paper's Mix'n'Match scheme introduces the ability to have three modules on each of two layers to be combined for these nine resulting styles. As more modules on these layers are considered, and as multiple modules on the other SRM-IMMPSs components are considered as well, the number of different potential applications grows rapidly with the introduction of new module instances.

The address information for each building stored in the Fiets document enables presentations that convey how the buildings on the Herengracht change as their addresses change. Figure 3 shows one means of conveying this, using the spatial layout of the presentation screen. Fiets demonstrates how the timeline or navigational interface of the final presentation could also be used to convey this same concept [21].

Processing in the design layer determines whether the "by address" concept described in the content layer module is to be conveyed using the presentation's layout, timeline or navigational interface. The encoding of each of these three means is made in a separate module that fits in the design layer. The "using layout" design layer module generates the presentation shown in Figure 3. If the "using timeline" design layer module is used instead, each building is shown on a full window one at a time for several seconds, in order of street address. Alternatively, if the "by year" design layer module is used with the "using timeline" design layer module, the buildings are shown one at a time in the order of the year in which they were built.

Also decided at the design layer is the use of text labels and the background imagery. These decisions are encoded in design layer modules. One design layer module may specify a 3x4 display of thumbnails with black-and-white canal building images taken from old sketches, as shown in Figure 3. A different design layer module may specify a 2x3 thumbnail layout with no background imagery.

Design layer style module DSSSL code could exist on the Web as one artist's concept of how documents of a particular document set should be presented. The data structure returned by the content layer style module DSSSL code described above is processed by the design layer. The design layer style module encodes how the order represented in the data structure is to be mapped to the final presentation, whether it be to the spatial screen layout, the timeline, or a list of ordered links. Code defining this processing is provided in the initial work on Berlage [23].

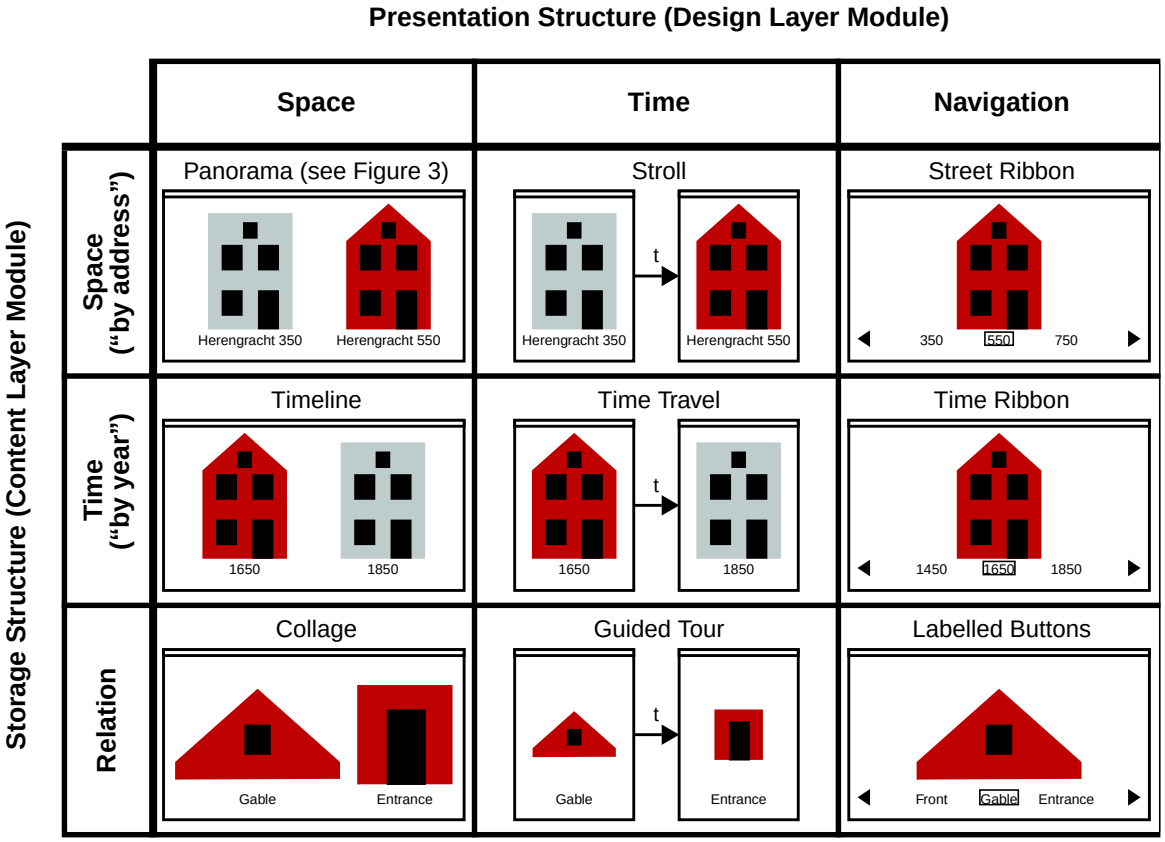

Figure 5: Fiets Structural Transformation Combinations [21]

Figure 6 shows DSSSL code for generating a Fiets presentation that focuses on the use of timing. It defines "seq" (sequence) as the primary SMIL construct to use for composing the presentation components, causing them to be displayed in sequence. The "make-regions" function, called from the control layer main Fiets DSSSL style sheet file, establishes one screen area for a building image and another for a descriptive caption. The "make-building-display" function calls functions defined in the SMIL application expert DSSSL file for displaying for each building its image and its descriptive caption in these regions. This function is also called from the main style file to generate the display SMIL elements for each building in a list of buildings, the order of which is determined by processing the constants assigned in the content layer module.

The DSSSL file for the design layer module that focuses on the use of space defines "par" (parallel) as the main SMIL composite, causes the buildings to be shown at the same time. It defines the make-regions function as defining an ordered table of image and caption regions.like that shown in Figure 3. Its make-building-display function places each building's image and caption in the regions appropriate for that buildings place in the ordered list determined by the content layer module used. Which aspect of the presentation structure is focused on is determined by which design layer module is included with the main Fiets style sheet when it is loaded into Jade for processing.

True exchangeability between the design and realization layers requires that the design layer output be appropriate for translation to a variety of hypermedia presentation formats. The existence of such a model is discussed in earlier work [25][26]. This earlier work also discusses how existing standards and tools can be adapted to encode and process this model.

```
(define DesComposite "seq")
(define (make-regions)
  (make sequence
    (make empty-element gi: "region"
        attributes: (list
          (list "id"     "text-region" )
          (list "height" "8%"          )
          (list "width"  "50%"         )))
    (make empty-element gi: "region"
          attributes: (list
            (list "id"     "image-region")
            (list "top"    "8%"          )))))
(define (make-building-display)
  (make sequence
    (make-label "text-region"
      (string-append (get-axisname)
        (get-number) ".txt"))
    (make-img "image-region" "2.000s"
      (entity-system-id
        (attribute-string "entity")))))
```

Figure 6: Example of Fiets Design Layer Style Module DSSSL Code for Time-based Presentation

The AHM provides a model for describing the basic spatio-temporal and navigational layout in a manner that is independent of the final output format [15]. This model could be used as the basis for communication between the design and realization layers. Other work provides a more detailed spatial layout model and the means of generating it in context of the SRM-IMMPSs [12]. This spatial layout model can also be part of what is communicated between the design and realization layers.

REALIZATION LAYER

The realization layer translates the desires expressed by the design layer into a directly playable hypermedia format. This layer determines exactly what spatial coordinates and what timing can match the constraints given by the design layer. If no such detailed specifications are possible, it then communicates with the design layer to determine an acceptable alternative. Similar communication occurs when the initial desires of the design layer cannot be expressed in the directly playable hypermedia format. The two layers then decide what acceptable alternative can be expressed in the output format.

The realization layer outputs code in some hypermedia format. One realization layer module may generate the design layer's specifications as a SMIL presentation. A different realization layer module may take the same specification and generate a presentation in the hypermedia presentation format MHEG [20]. The Berlage component that best corresponds with the realization layer is the generated SMIL code itself. This acts as an embodiment of the presentation that can be played on a number of platforms by a number of systems.

AHM constructs that apply to realization layer processing are channels. They define alternatives for presenting document components that can be selected from. A realization layer style module can encode the selection of one channel-defined alternative in determining the final presentation encoding.

PRESENTATION DISPLAY LAYER AND THE USER

The presentation display layer is embodied by the particular player software that is called at presentation time to process a segment of code generated by the realization layer for presentation to the user. It also handles the user's interaction with the generated presentation. Because the presentation is fixed at this point, there are no style modules for this layer.

The GR*i*NS player in the Berlage environment corresponds with the presentation display layer. Here, GR*i*NS plays the SMIL code passed to it by the realization layer. It also processes the user's interaction with the SMIL presentations shown, as described in earlier work on the introduction of dynamics to Berlage [22]. In processing user interaction, GR*i*NS acts as an http client, sending each user-activated http request to the http server in the Berlage environment. Each user interaction activates each step in the presentation process. With each interaction, the style sheet, with the

newly updated presentation status DSSSL code it includes, is processed to generated the next step of the presentation to the user.

APPLICATION AND APPLICATION EXPERT

The application component provides access to the stored hypermedia information that is presented to the user. This includes the individual media objects presented, the metadata around these media objects and the general facts concepts and get conveyed to the user. All of this data is stored in a variety of formats, locations and systems

The application expert provides an interface for the various systems and formats that are involved in generating the presentation. This includes the formats of both the media that is feed into the generation process and the media that is generated as part of the generation process for inclusion in the final presentation. A separate application expert module could exist for the input and output of each format used in a generated presentation. With this model, any number of application expert modules could be used simultaneously for the generation of a single presentation.

The application expert can also provide access to the format in which archival hypermedia information representing more general concepts and facts is stored. This represents the knowledge of a particular semantic domain. The application expert provides the generation process access to this information and enables it to be transformed into presentation to the user.

In Fiets, text content is often converted into image files that display the text. This enables the constructs in SMIL 1.0 to have the control desired for laying out this text with other visual media. As described earlier, the content layer determines what characteristics, such as font size, the text should have given the user's characteristics, as encoded in the user expert module, and perhaps information from other sources as well. Once the content layer determines how this text should look, it then passes these specifications to the application expert to create the appropriate image file in a image format such as JPEG.

The components of the Berlage environment architecture that correspond to the application in the SRM-IMMPSs are the SGML and HyTime encoded files. These include primarily the document, which encodes the knowledge presented and the metadata around the media objects used in the presentation. The application also includes the media objects and the systems that store and provide access to them. These media-specific aspects of the application are not represented explicitly in the Berlage environment architecture. Their processing is handled by GR*i*NS, which is the only Berlage component that processes the media objects directly.

Some Berlage components that act as application expert style module are the HyTime, Berlage architecture and document set (Fiets) DSSSL libraries. Each of these is a module providing access to semantics encoded in the context of a particular document set. Each of these document sets corresponds with a particular meta-DTD or DTD. Code in these modules defines how the properties represented in the syntax of these document sets can be queried against by the style sheet. Earlier work on Berlage provides example code defining the access to HyTime properties used to represent the street addresses of buildings [23].

The Berlage SMIL DSSSL library also acts as an application expert module since it provides the system information it needs about a particular format. What distinguishes this application expert style module from the others is that it enables the *output* of a format rather than its input. The SMIL DSSSL library provides functions that can be called by the design layer style module. These functions enable the design layer style module to specify its output in terms of more general hypermedia constructs instead of specific element and attribute assignments. The SMIL DSSSL library acting as the application expert style module turns these functions calls into SMIL output.

USER EXPERT

The user expert provides information about the user that is processed to adapt the presentation to the user's characteristics. These characteristics include abilities, preferences and areas and levels of expertise. A style module for this layer would encode this information about a user. One style of presentation can be tailored for different users by switching only the user expert modules. One user could have such a module that could be plugged into any presentation generating style.

In the Fiets example being discussed, the user expert module would inform the system of the user's level of visual acuity. This information could then be used to, among other things, determine what size font of text to use, such as with the text shown in Figure 3. Design layer processing would communicate with the user expert to output the text with the appropriate font.

The user expert has the potential to effect more than design layer processing. In the Fiets example, the realization layer processes the layout requirements of text generated by the content layer given the user's requirements. If the space required for this text is too large for the layout specifications from the design layer, the realization layer may then communicate with the design layer instructing it to change the layout. so that each screen is less crowded.

DESIGN EXPERT

The design expert communicates the constraints of the final presentation environment. This includes primarily information about the platform on which the presentation is rendered, such as what media peripherals are available.

One commonly occurring platform constraint in generating multimedia presentations is that most personal computers can only play one sound file at a time. This type of information is most appropriately encoded in the design expert mod-

ule. When the generation process knows from this module that only one sound file can be played at a time in the final presentation, the realization layer module can be encoded to ensure that no two sound files are ever scheduled for simultaneous playback. Furthermore, the design layer module can be encoded to combine multiple sound files into a single file when simultaneous playing of them is desired on a restricted system.

No particular component of Berlage as illustrated in Figure 2 can be singled as corresponding with the either the user expert style module or the design expert style module. The DSSSL code files that would be included by reference from the main style sheet would define constants that describe the abilities and preferences of the user and the facilities of the presentation system.

CONTEXT EXPERT

The context expert keeps track of the activity that has occurred during a given presentation. This information is important to the content layer in determining whether goals and subgoals have been met and in determining what future actions are required to complete them. There is no pre-written module required for this expert because the information it provides is generated automatically during the presentation itself. Also, the processing of this information is specified for the most part by the content layer module.

The context expert is represented in Berlage by the presentation status DSSSL code. This code contains information about what was presented to the user and how the user reacted to it. It is a log file that is augmented with each transmission of a SMIL file to GR*i*NS and each link activation by the user. It is processed with the DSSSL code to generated each step of the presentation.

CONCLUSION

This paper discusses how the generation of a hypermedia presentation from stored media data can be encoded as a style specification that is broken up in distinct modules. The components of the SRM-IMMPSs are used as the basis for defining the scopes of these modules. This gives the user more control over the style of the final presentation of a document because rather than selecting one complete style, the user can plug in the most appropriate module for each aspect of style.

The modular division this paper describes provides the basis for defining style specification that can be interchanged between different presentations. Aspects of style are isolated that can be applied uniformly to the generation of multiple presentations from multiple sources. This enables future conventions and standards for style to be defined in terms of these modules, which facilities information exchange and reuse.

The potential implementation of the Mix'n'Match technique is described in terms of the Berlage environment architecture. This description provides the basis for the wide-spread implementation of Mix'n'Match style module usage with public standards and tools for the Web. It can also act as a guide for the development of future standards to facilitate the use of modularized style on the Web.

One issue that remains to be address is how broad of a document topic domain can be covered by the same module instances. Can one design module instance apply to, for example, both Fiets and the design of airplane components? The answer to this question lies in experimenting with Mix'n'Match-like environments that work for broader document domains. Also remaining to be addressed is what the nature of authoring practice would be in a Mix'n'Match-like environment. What issues are raised by authoring a single component of style rather than an entire presentation? Can an environment be made in which authors trust that the aesthetic intentions encode state in their modules will not be overridden by other modules during processing? These questions would be answered by further use of environments such as the one described in this paper.

ACKNOWLEDGMENTS

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

REFERENCES

1. City of Amsterdam Municipal Department for Preservation and Restoration of Historic Buildings and Sites. Amsterdam Heritage, URL: http://www.amsterdam.nl/bmz/adam/adam.html.

2. 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), pp. 477-496.

3. Bray, T., Paoli, J. and Sperberg-McQueen, C.M., Extensible Markup Language (XML), W3C Recommendation (January 1998), URL: http://www.w3.org/TR/REC-xml.html.

4. 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 (Melbourne, Australia, April 1998).

5. Bulterman, D.C.A., Rutledge, L. Hardman, L and van Ossenbruggen, J. Supporting Adaptive and Adaptable Presentation Semantics, in Proc. The 8th IFIP 2.6 Working Conference on Database Semantics (Rotorua, New Zealand, January 5-8, 1999).

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

7. Clark, J. SP — An SGML System Conforming to International Standard ISO 8879 — Standard Generalized Markup Language, URL: http://www.jclark.com/sp/.

8. Clark, J. XP — An XML Parser in Java, URL: http://www.jclark.com/xml/xp/.

9. CWI (Centrum voor Wiskunde en Informatica). GR*i*NS — The GRaphical *i*Nterface to SMIL, URL: http://www.cwi.nl/GRiNS/.

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

11. Goldfarb, C. The SGML Handbook. Oxford University Press, 1991.

12. Graf, W.H. Intelligent multimedia layout: a reference architecture for the constraint-based layout of multimedia presentations. Computer Standards and Interfaces 18(6,7) (December 1997), pp. 515-524.

13. Halasz, F., and Schwartz, M. "The Dexter Hypertext Reference Model". Communications of the ACM. Vol. 37, No. 2, February 1994.

14. Hardman, L., Bulterman, D.C.A., and van Rossum, G. "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model". Communications of the ACM. Vol. 37, No. 2, February 1994.

15. Hardman, L., Worring, M. and Bulterman, D.C.A. Integrating the Amsterdam hypermedia model with the standard reference model for intelligent multimedia presentations. Computer Standards and Interfaces 18(6,7) (December 1997), pp. 497-507.

16. Hoschka, P. (ed.). Synchronized Multimedia Integration Language, World Wide Web Consortium Recommendation. June 1998. URL: http://www.w3.org/TR/REC-smil/.

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

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

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

20. International Standards Organization. Coding of multimedia and hypermedia information — Part 5: Support for base-level interactive applications (MHEG-5), ISO/IEC IS 13522-5:1997. International Standards Organization, 1997.

21. Rutledge, L. Hardman, L., van Ossenbruggen, J. and Bulterman, D.C.A. Structural Distinctions Between Hypermedia Storage and Presentation, in Proc. ACM Multimedia 98 (Bristol, England, September 12-16, 1998), pp. 145-150.

22. Rutledge, L., Hardman, L., van Ossenbruggen, J. and Bulterman, D.C.A. Implementing Adaptability in the Standard Reference Model for Intelligent Multimedia Presentation Systems, in Proc. Multimedia Modeling 98 (Lausanne, Switzerland, October 12-15, 1998), pp. 12-19.

23. Rutledge, L., van Ossenbruggen, J., Hardman, L. and Bulterman, D. Practical Application of Existing Hypermedia Standards and Tools, in Proc. Digital Libraries 98 (Pittsburgh, USA, June 23-26, 1998), pp. 191-199.

24. Rutledge, L., van Ossenbruggen, J., Hardman, L. and Bulterman, D. A Framework for Generating Adaptable Hypermedia Documents, in Proc. ACM Multimedia 97 (Seattle, USA, November 9-13, 1997), pp. 121-130.

25. Rutledge, L., van Ossenbruggen, J., Hardman, L. and Bulterman, D.C.A. Generic Hypermedia Structure and Presentation Specification, in Proc. Electronic Publishing 97 (Canterbury, England, April 14-16, 1997), pp. 177-187.

26. Van Ossenbruggen, J., Hardman, L. Rutledge, L., and Eliëns, A. Style Sheet Support for Hypermedia Documents, in Proc ACM Hypertext 97. (Southampton, England, April 6-11, 1997), pp. 216-217.