Rutledge, L., van Ossenbruggen, J., Hardman, L., Bulterman, D.C.A., and Eliëns, A., "Generic Hypermedia Structure and Presentation Specification", *Proceedings of ICCC/IFIP Conference on Electronic Publishing*, April 1997.

Generic Hypermedia Structure and Presentation Specification

Lloyd Rutledge[*], Jacco van Ossenbruggen[†], Lynda Hardman[*], Dick C.A. Bulterman[*], and Anton Eliëns[†]

[*]CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
lloyd@cwi.nl
lynda@cwi.nl
dcab@cwi.nl

†Vrije Universiteit
Fac. of Mathematics and Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
jrvosse@cs.vu.nl
eliens@cs.vu.nl

ABSTRACT

We consider the generic hypermedia structure of a document to be a means of representing the document that allows it to be processed into a wide variety of presentations. Representing a document in this manner requires additional specification and resources to render it into any presentation. In this paper we discuss the relationship between the generic hypermedia structure of documents and the processing of this structure into presentation. Our discussion is expressed in terms of existing models for hypertext and hypermedia systems and also in terms of ISO standards for text and hypermedia document formatting and processing. The discussion and the resulting formalisms and the resulting formalisms are illustrated with extension designs for the hypermedia authoring and presentation environment developed at our laboratory.

Keywords and Phrases: Hypermedia, Document Processing, Dexter, AHM, SGML, DSSSL, HyTime

1. Introduction

Our research group has developed the CWI Multimedia Interchange Format (CMIF), and the CMIFed environment for the creation and playback of CMIF hypermedia documents[4]. CMIFed is implemented for a variety of platforms, making CMIF-encoded documents portable for display to these platforms. CMIF and CMIFed incorporate facilities characteristic of current state-of-the-art hypermedia systems.

The Dexter Hypertext Reference Model specifies significant abstractions shared by typical hypertext systems to provide a common reference for discussing and comparing these systems[3]. It is designed to aid in the general discussion of different hypermedia documents and applications. It defines a layered model onto which individual hypertext systems can be mapped. It also defines commons terms to be uniformly applied to the hypertext systems when they are described by Dexter.

Our research group has developed the Amsterdam Hypermedia Model (AHM) for hypermedia documents[4]. The Amsterdam Hypermedia Model (AHM) extends the Dexter model into multimedia by adding to it components of the CMIF hypermedia document model. These extension include the synchronization of multimedia object display and more complex specifications of how link activation affects document presentation.

Standard Generalized Markup Language (SGML)[6,2] is a widely used ISO standard for representing electronic text documents. It supports and facilitates document exchange, reuse and longevity by representing a document in its permanently stored form without defining its processing into any particular presentation. SGML also provides for the definition of separate document sets, called *Document Type Definitions (DTDs)*, which represent different types of documents used by separate communities. An SGML document conforming to a particular DTD can be processed by any system designed for that DTD. SGML provides a basic set of syntactic and hierarchical constructs for text. The placing of semantics upon SGML constructs is outside the scope of the standard and is considered in the province of applications processing a DTD. Individual DTDs place their own semantics on their SGML construct composites.

Hypermedia/Time-based Structuring Language (HyTime)[5,1] builds upon and extends SGML presentation-independent structuring from text into hypermedia. This additional structure includes complex hyperlinking, locating of document objects, and the scheduling of objects within measured coordinate systems such as space and time. HyTime inherits SGML's ability to define distinct document sets. Portability and reuse of information is further enhanced beyond that of SGML by allowing HyTime documents to include portions of other documents, even ones from other DTDs. HyTime, like SGML, requires external mechanisms for specifying the presentation of its document.

We have developed a HyTime DTD for CMIF, enabling any CMIF document to be translated into an instance of this DTD for incorporation into other HyTime systems. This translation provides an empirical test of HyTime's ability to represent the hypermedia structure of existing environments. It also provides an opportunity to explore the issues behind processing this generic structure for actual presentation.

This paper presents a unified perspective of hypermedia processing derived from the models and standards discussed. It also presents extensions to our hypermedia system based on this perspective. We begin with discussing these models and standards and the unified perspective they imply. Next, we describe generic hypermedia structure and presentation specification in more detail, based on this unified perspective. Finally, to illustrate these discussions, we present extensions to our CMIFed environment that utilize HyTime and the presentation specification issues presented in this paper.

2. A Unified Perspective of Generic Hypermedia Structure

2.1. Generic Hypermedia Structure

The Dexter model and its relation to HyTime is illustrated in Figure 1. Dexter divides hypertext processing into three layers: the *run-time*, *storage*, and *within-component* layers. The run-time layer is concerned with the final presentation of documents, including how the user interacts with the interface and what activities occur during presentation. This layer makes up the user's perception of the document. The within-component layer contains the *nodes*, or the atomic containers of the media objects inside a document. This layer represents the individual items contained in the document. The storage layer provides the hypertext structure of a document. The storage layer is related to the within-component layer through *anchoring*, which enables media type-independent linking to media content. The *presentation specification layer* defines how the contents of the storage layer are to be processed when the user browses through them. The Dexter model focuses on the storage layer and its connections with each of the other two layers, maintaining independence from particular presentation applications and source media formats. This layer corresponds with what we call the *generic structure* of the document.

The basic building block of the storage layer is the *component*. The *atomic component* wraps the actual media data constituting the hypermedia document. The *composite component* allows for hierarchical structuring of these data. The *link component* is used to define hyperlinks among atomic, composite and other link components. The

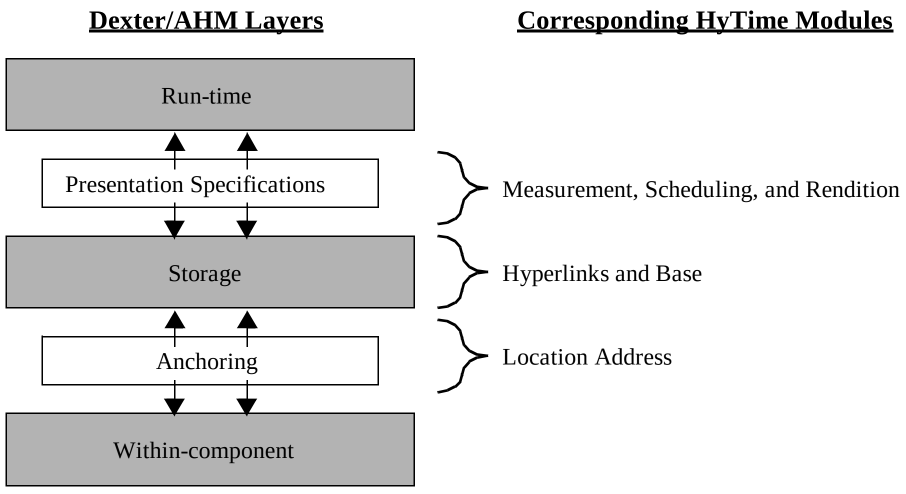
Dexter/AHM Layers
Corresponding HyTime Modules
Run-time
Presentation Specifications
Storage
Anchoring
Within-component
Measurement, Scheduling, and Rendition
Hyperlinks and Base
Location Address

Figure 1. The Dexter/AHM Layers and Corresponding HyTime Modules

structure as described by the components in the storage layer is independent of the actual type of media data used and the details describing its presentation. The storage layer defines the overall structure of the document itself, independently of the document's presentation or the storage of its contents. As such, the Dexter model separates content, structure and presentation.

Dexter allows for multimedia data within its atomic components. However, it does not qualify as a hypermedia model since it does not support temporal composition and synchronization[3]. The Amsterdam Hypermedia Model (AHM) extends the Dexter model storage layer by introducing composites for defining synchronization that are based on temporal and atemporal composition. CMIF channels are represented, which provide independently controlled presentation of document components. Further, the extension can describe how the context of a link's activation is to be affected by the link's traversal. These extensions provide the Dexter model with more constructs for the document's rendered presentation and for generating this presentation from the storage layer representation.

In SGML terminology, a document is separated into *content*, *structure* and *presentation*. The *content* makes up the individual perceivable components of the document. The *presentation* is who the document appears to the user. SGML focuses on document *structure*, which specifies the relations between the content components that affect how they may be presented without specifying any particular means of presenting them. Roughly, content corresponds with the Dexter within-component layer, structure with the storage layer, and presentation with the run-time layer.

The HyTime base module provides the building blocks for general document structure that are typically components of the more semantically specific constructs from the other modules. These building blocks by themselves can define some linear and hierarchical structuring beyond that provided by SGML. These structures correspond with the composite structures of the Dexter storage layer. The HyTime hyperlinks module establishes non-hierarchical and non-linear relationships between document components. The structures defined by this module correspond very closely with links in the Dexter/AHM storage layer.

3. A Unified Perspective of Hypermedia Presentation Specification

A document's generic hypermedia structure cannot by itself be presented to a user. This structure must accompany a mapping that specifies how that structure is translated into a particular presentation. This mapping references aspects of the document's generic hypermedia structure to determine the document's appearance to the user. The mapping may also reference aspects of the document's presentation environment to determine the best means of displaying the document under the conditions given.

Below we describe several different areas of presentation specification. We then discuss the approaches the models, standards, and implementations discussed in the paper take towards presentation specification. Finally, we unify these different approaches to provide a uniform and cooperative view. This unified view includes some ideas for extending DSSSL to work with HyTime and other hypermedia documents.

3.1. Areas of Presentation Specification

We identify five areas of presentation specification: *appearance*, *navigational interface*, *render-time characteristics*, *presentation-time characteristics*, and *user characteristics*. These areas are described below.

3.1.1 Appearance

The appearance of a hypermedia document is the collection of presentation characteristics that do not involve navigation, other run-time determine properties, or any properties that are determined after authoring time. Static appearance properties include background color, font type and size, and justification. These are characteristics that can be authored into a style sheet at the same time the accompanying document is authored.

3.1.2 Navigational Interface

For a given hypermedia document the user's control over its presentation can vary. Aspects of the navigational interface include what appears to the user at a given time, how available and activated navigational choices are made perceivable by the user, and what the potential traversal paths through the document are. It is possible that structural components that bring about navigational choices in one presentation of a document may not do so in other presentations of the same document.

3.1.3 Render-time Characteristics

Some characteristics that can affect a document's presentation can change after the document itself is authored. The generation of a document's presentation, in order to be up to date, would have to occur after all such characteristics have been altered. For example, if a document contains two video clips presented in succession, the time at which the second video starts depends on the length of the first video. This document may be authored to refer to the file system locations of the two videos. If the video clip in the first file changes, so might its length, and thus so would the time at which the second is displayed.

3.1.4 Presentation-time Characteristics

Other aspects that affect a document's presentation are determinable only at the time the document is presented. One example is available resources and communication bandwidth on the presentation environment at the time the document is displayed. The document's presentation specifications may take into account quality of service issues, and these are only determinable at the time of presentation. Also, certain aspects of presentation timing can only be determined at presentation-time, such as the termination of a live video feed, or a signal given by an external process. Generically structured documents may need to make references to such events so that the proper conclusions regarding timing can be made during presentation.

3.1.5 User Characteristics

Tailoring a document's presentation for a particular user improves the efficiency with which the user obtains the desired information. User characteristics can include expertise in a particular area, which may determine what information is unnecessary to present, and the user's abilities, which can provide for using alternative media types for media objects the user may not be able to perceive.

3.2. The DSSSL Approach

Since SGML and HyTime documents only describe generic, presentation-independent structure, they generally need further processing to be indexed, formatted, and printed or displayed. DSSSL[7] specifies in a platform-independent way how an SGML document or document set should be processed. To achieve this, DSSSL defines two independent processes. The first is the *transformation* process, which is used to convert a document conforming to one SGML DTD to a document conforming to another. Such transformations are used, for example, to convert company specific documents to standard conforming ones, or for translation of a complex and rich document to a more easily processed one (e.g. conversion from a domain specific document to HTML). The second process is the *formatting* process. The document describing the formatting process is traditionally called style sheet. To be able to develop platform independent style sheets, DSSSL introduces an abstract target representation to convert to. This abstraction, the flow object tree, is a flow-based model of a sequence of pages. Typical objects in the tree are column, header and footer objects, paragraph objects, anchor objects etc. All objects have a specific set of attributes which contain additional information for the formatting application.

A typical DSSSL style sheet is a platform independent program which specifies the mapping from SGML elements to objects in the flow object tree in a very declarative way. It is up to a back-end application to convert the flow object tree to the final presentation form (typically RTF, PostScript or PDF). This back-end needs to implements justification, line breaking and page breaking algorithms. Note that the output of the formatting process is a physical description of the document, and all information about the logical structure of the document is lost. As a result, the formatting process can in general not be used for conversion to other structured, but non-SGML encodings (such as LaTeX, or, in our case, CMIF).

3.3. The Dexter/AHM Approach

To specify the way components are to be presented, Dexter introduces *presentation specifications (pspecs)* as an interface mechanism between the storage and run-time layer. While the structure of a pspec is not defined in the Dexter model, they play an important role. First, atomic components have a pspec to be able to store details about the way they need to be presented. Composites have their own pspecs, though whether these override the pspecs of their children or vice versa is not specified. Links also have pspecs to describe their presentation. Finally, a pspec is used in the run-time layer to allow end-users to influence the way a component is presented.

The Amsterdam Hypermedia Model introduces temporal composition by the composite type in the definition of the composite component. More detailed synchronization is defined in the presentation specification of the atomic and composite component. The pspec of an AHM atomic component specifies a duration of the multimedia data it contains. The pspec of a composite, called a *synchronization arc*, contains the set of temporal constraints between any of its children.

3.4. The HyTime Approach

Though most of HyTime's constructs contribute to what in this paper is called generic structure, a few perform presentation specification functionality. These include the concepts of accessed hyperlinks, the containment of access anchor lists in schedules, and portions of the measurement, scheduling, and rendition modules.

3.4.1 HyTime's Focus Away from Presentation Specification

Both Dexter/AHM and HyTime keep their focus away from document presentation to enable presentation-independence, but HyTime's lack of specification for this area is more pronounced. HyTime does not provide many facilities that Dexter/AHM does for guiding a document's rendering to a presentation based on information that is available only after authoring, such as render-time, run-time, and user-characteristics. For example, while HyTime's dimension referencing can be done in terms of other concretely placed objects or in terms of mathematical formulas, it does not provide for such referencing to be done in terms of object dimensions that are only determinable at run-time. An example of such run-time referencing is having one MHEG video clip start when another ends when the length of the clips are not known during authoring. Some aspects of HyTime that do contribute to presentation specification are described in the following sections.

3.4.2 Measurement Module

The HyTime measurement module specifies the use of numeric measurement in hypermedia structure. Its constructs are not used by themselves but in conjunction with those of other modules. The *marker* construct defines a single number to be used as one aspect of an objects measurement: either its starting point, its ending point, or its duration or length.

3.4.3 Scheduling Modules

The presentation of hypermedia documents typically involves the use of coordinated measurements, such as the positioning of images at locations on a screen and at particular moments along a timeline. The scheduling module defines the constructs that specify such coordinate systems. The HyTime *finite coordinate space (fcs)* defines what axes such a coordinate system can have, and what units of measure are applied to them. A HyTime *event schedule (evsched)* specifies one instance of a particular fcs. A HyTime *event* represents one measured area of a schedule. An event specifies this area as a pair of coordinates for each axis. Event contents, typically one media object, are considered positioned at those coordinates in the schedule.

Due to the numerical and measured nature of schedules, their structure typically has a more direct impact on presentation than compositional or hyperlink structure. However, the specific numbers use in the schedules are defined by measurement module constructs, leaving the structure defined by the scheduling module to be less numeric and more compository in nature, and thus more generic. Further, although a HyTime schedule can define presentation spaces such a 2-dimensional screen displays with timelines, presentation-independent schedules, such as a geographic schedule for countries of the world measured in longitude and latitude, can exist as well.

Such schedules are generic hypermedia structure and require style sheet mappings to specify their presentation to a user.

3.4.4 Rendition Module

The module that, in its entirety, has the most potential to contribute to a document's presentation specification is the rendition module. It defines how the structure of one schedule and its contents is derived from that of another. This functionality lends itself readily to defining how the schedule specifying a particular presentation is derived from a more general schedule. Such a specification could, in fact, be written as a document separate from the source schedule, and in such a case would serve more as a style sheet than a generically structured document. However, rendition can also specify mappings between two presentation-independent structures, and such mappings would not qualify as presentation specification.

3.4.5 Accessed Anchors of Hyperlinks

Though there is no specific construct associated with it, HyTime defines the concept of an *accessed hyperlink anchor*. According to the HyTime general model, each hyperlink has a collection of anchors, and traversal can occur through the hyperlink from one of its anchors to another. An anchor that is traversed to in this fashion is considered *accessed*. Since traversal an access happen only during document processing or presentation, aspects of HyTime structuring that relate to the accessing of anchor fall in the area of presentation specification. HyTime does not specify what makes an anchor accessed. This is left up to applications that use HyTime processing to decide in each case.

One use in HyTime of accessed anchors that is particularly applicable to presentation specification is the *accessed anchor list (accanch)* construct. An accanch specifies a collection of anchors as potential objects to be contained in a schedule event. The objects from this collection that are considered positioned in that schedule area at any given moment in runtime are those that are considered accessed through a hyperlink at that moment. Thus, the HyTime-defined structure itself can be changed during the course of the document's processing or presentation. Furthermore, how this structure changed during presentation is itself defined by HyTime.

3.5. Potential Extensions to DSSSL

DSSSL is designed for the processing of text documents. DSSSL does not support the scheduling and synchronization of time-based hypermedia fragments such as audio, video and animation. Using DSSSL for hypermedia documents would require extensions to DSSSL's output model to be able to output hypermedia presentations and extensions to give the style sheet access to information which is only available at presentation time.

3.5.1 Hypermedia Output Model

An advantage of DSSSL is the platform independent and declarative nature of the style sheets describing a formatting process. DSSSL stylesheets describe formatting in terms of an abstract page model as described by the flow object tree. A drawback of this approach is that since the flow object tree models a two dimensional, physical page representation of the document, DSSSL is only suited for page-based destination formats. As a result, multimedia documents can not be represented by DSSSL's flow object tree because it lacks a temporal dimension. There are two ways to avoid the limitations of DSSSL's two dimensional page-based output model.

First, one could only use the transformation process, which does not use the flow object tree. Because this process involves SGML to SGML transformation it requires that the physical representation to be encoded in SGML. This is a slightly counter intuitive approach, since SGML was designed for declarative, generic markup.

The second approach is to use the formatting process, despite the limitations of its two-dimensional flow object model. DSSSL's set of flow objects can be extended for a particular application domain. New objects can be introduced to model multimedia specific features, such as spatio-temporal composition, synchronization and user interaction. From a modeling point of view, however, the question remains how well the new multimedia specific objects can be integrated into the existing ones provided by the DSSSL standard.

There are some practical limitations as well, since this approach would require a non trivial extension of the DSSSL back-ends used, and the document can no longer be processed on other (standard) DSSSL systems.

3.5.2 Access to Presentation-time information

DSSSL provides hooks to allow involvement of external processes in the transformation and formatting process. Such processes may provide the information which is only available at run-time. Typical examples include information about the available (network) resources, the result of QoS negotiation, run-time measurement information and user characteristics. Again, the introduction of external processing would make the DSSSL stylesheet more platform dependent. However, not all run-time processing needs to be in the style sheet. The style sheet merely needs to provide sufficient information to the playing environment so it will be able to generate an optimal presentation of the hypermedia document based on that information. Actual processing of that information can be done in the playing environment.

So while DSSSL is designed for static, page-based output and not for dynamic and interactive multimedia or hypermedia documents, there are several advantages in using DSSSL for hypermedia as well. First, even dynamic hypermedia documents need some formatting of static text and graphics. DSSSL already provides facilities to do this, and these facilities can be very well reused in a hypermedia environment. Furthermore, DSSSL offers an expressive query language which allows querying of hypermedia documents based on both content and structure. A style sheets could use the query language to retrieve specific parts of the document based on the run-time information described above. With DSSSL’s HyTime support, queries can use the powerful locating constructs defined by HyTime to find specific parts of the document.

4. Using HyTime and Presentation Specification with CMIF

To enhance the portability of CMIF documents and the information they contain we have developed a HyTime-encoding for CMIF and a translator from CMIF to this encoding. This format is represented by a DTD for a HyTime-conforming document set. Figure 2 contains a diagram representing this DTD. The text version of the DTD is available through a WWW page[8]. We are also developing presentation mechanisms for documents represented by this HyTime encoding. We discuss this HyTime-defined document model and these presentation mechanisms in this section.

Our approach has been to encode as much of the CMIF structure as possible in HyTime, although some of the structure in CMIF is presentation-oriented. One motivation for this is to ensure that the document has a default presentation in the absence of accompanying style sheets. Another is to allow this information to be used as over-ridable guidelines in generating the actual presentation. For example, much of this structural information includes explicit screen and timeline positioning. CMIF documents derive information about timing from this structure, and by preserving in HyTime as much of this original CMIF information as possible, calculation of the actual timing and presentation information is allowed at playback time. This allows flexibility in playing back presentations with non-predictable timing environments (e.g. network delays) and for different end-user platforms.

Despite our HyTime encoding of presentation, we have taken steps to enforce the separation of generic structure and presentation specification. First, this the portions of the structure that fit in different layers of the Dexter/AHM model are placed in distinct locations in the documents. This allows the more presentation-independent structuring from the lower layers, as shown in Figure 1 to be easily distinguishable from the more presentation-oriented higher layers. Second, this type of structural information is optional in our document model, so a valid HyTime-encoded CMIF document can exist with no presentation information in it. Finally, this information can be overridden by accompanying style sheets.

The CMIF document format is expressed using three different types of HyTime constructions. The first is the hierarchical and compository document structure, derived directly from the CMIF document. The second is a list of hyperlinks which record the link information stored in the CMIF document but also require the explicit specification of contexts derived by the CMIF player. The third is scheduling, which requires information from the CMIF document and durations of atomic nodes calculated by the player, and the presentation timing calculated by the player. The measurement and scheduling structures are the most complex parts of our HyTime DTD.

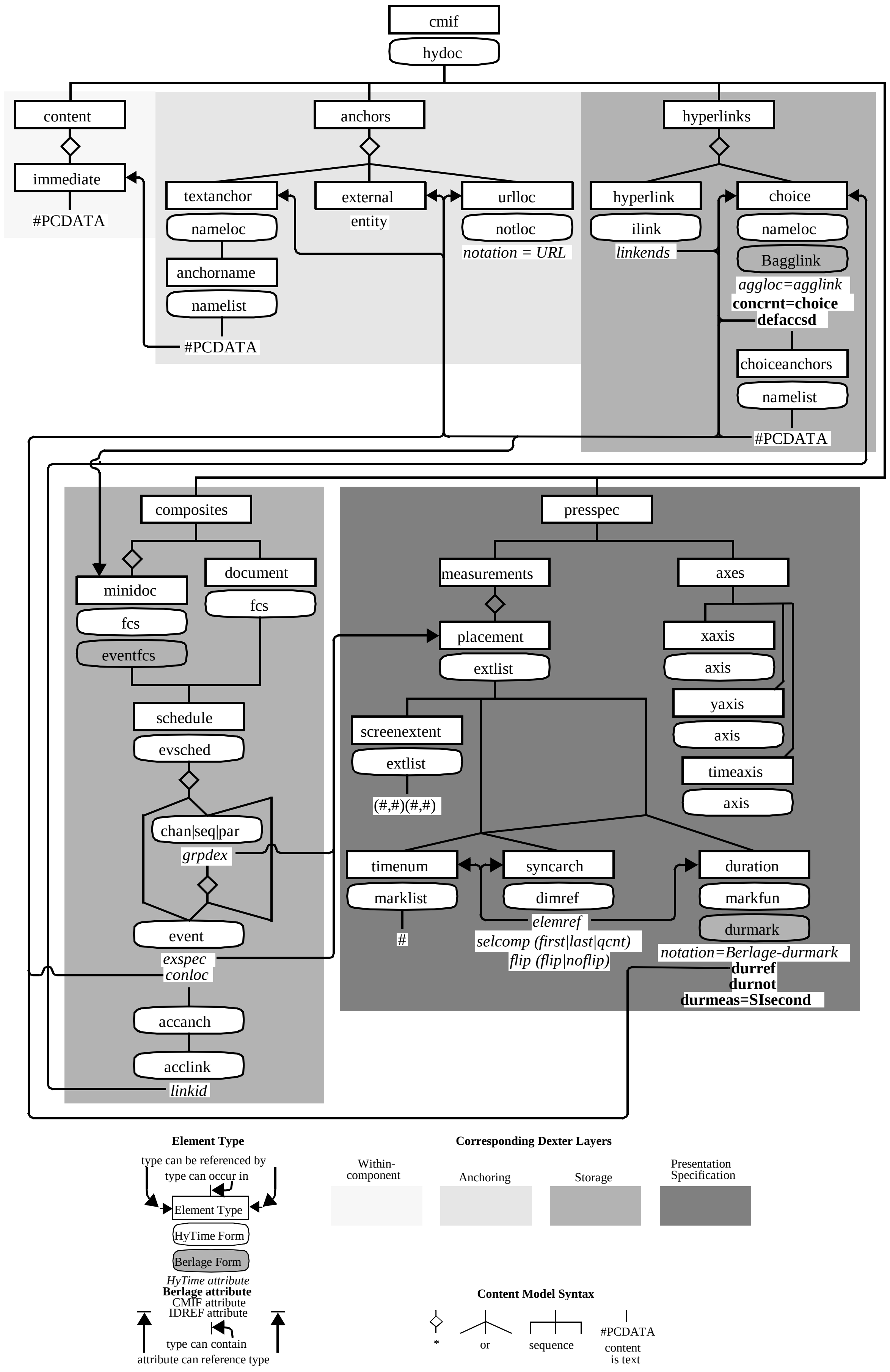

Figure 2: The HyTime DTD for CMIF

4.1. Architectural Extensions to HyTime for Presentation Specification

HyTime is defined not as a DTD but as a *meta-DTD*. This allows multiple DTDs to be defined for documents that use HyTime. A DTD specifies explicit element types, but the HyTime meta-DTD specifies collections of attributes, called *architectural forms*, that can exist in the element types of a DTD. If an element type is said to conform to an architectural form, then it uses the attributes defined for that form. Such an element type can use other attributes as well that fall outside of HyTime. This enables new document sets to be created that use HyTime constructs and incorporate their representational hypermedia semantics.

In writing a HyTime DTD for CMIF, we encountered some presentation specification concepts that were broad enough to apply to many hypermedia document sets. We have taken the same meta-DTD approach to define our own architectural forms to represent these presentation specification concepts in the same manner that HyTime's meta-DTD represents generic hypermedia structuring concepts. We call this meta-DTD and the forms it defines the *Berlage architecture*, named after the Dutch architect of several important buildings in Amsterdam. Through this meta-DTD approach, these presentation specification forms can be used in DTDs for other HyTime document sets. These new forms are defined with HyTime form attributes, and thus elements conforming to these presentation specification forms also conform to their HyTime forms and will be recognized as such by HyTime engines, even if the engine does not recognize forms of our new architecture. The Berlage architecture extends HyTime, rather than overriding it. Berlage documents are also HyTime-conforming documents. Further, semantics deemed appropriate for HyTime representation are encoded with HyTime constructs in the Berlage architecture. What Berlage provides are some presentation-specification layer constructs that style sheets can refer to in addition to HyTime generic structure constructs in determining a presentation.

Below are examples of some of the presentation specification forms we have defined for use with the CMIF DTD.

4.1.1 Render-time Markers

One characteristic of render-time presentation specification is specified by CMIF's *synchronization arcs*. With synchronization arcs, you can define the timing of one media object presentation in terms of another time event. Sometimes the resulting calculation involves the duration of a media object's presentation. Since most CMIF media objects are stored externally to the CMIF document itself, they are subject to modifications that are more recent that the final modification time of the document, and thus the duration of these media objects is not always certain at authoring time. The CMIFed environment, when processing a CMIF document, also processes these media object files to check their duration so that all timing information can be calculated for the document's presentation. Such processing is render-time. Duration synchronization arcs are thus author-time generated constructs that refer to render-time characteristics.

HyTime *dimension referencing* allows one marker to be defined in terms of another. The other marker can be an explicitly defined measurement or a dimension reference, allowing multiple levels of indirection. However, in HyTime, a dimension reference must eventually refer to either an explicitly stated number or a *marker function (markfun)*, which leaves the determination of the measurement to an unspecified external process. Our architectural extension has a *duration marker (durmark)* form, which extends a HyTime marker function to specify that it returns a number representing the duration or length of a media object. A duration marker has attributes that specify the media object itself, what its format is, and what unit of measure applies to the number returned.

This form is used with the duration element of the CMIF DTD, as shown in Figure 2. The duration element represents a synchronization arc that refers to the duration or length of a media object stored in an external file.

4.1.2 Unaccessing Anchors

The HyTime processing of some constructs is affected by whether or not a document component is considered accessed, as discussed in Section 3.4.5. This is useful information for the rendering of a document into presentation because the accessing of anchors specifies the changing of the presentable document structure. The HyTime standard states that traversal through hyperlinks causes anchors to become accessed. We extend this facility in the Berlage architecture to provide generic guidelines for having anchors become considered *unaccessed*.

The Berlage architecture has the form *Berlage aggregate link (Bagglink)*. It extends the HyTime aggregate link construct by specifying aspects of how subsequent access of the anchors in the aggregate affect the status of previously accessed anchors. For example, the value of the *concurrent (concurnt)* attribute of the form is "choice", then the element is a Berlage *choice aggregate*, and only one anchor in the aggregate is considered accessed at any time. When any of its anchors are accessed through a hyperlink traversal, any other previously accessed anchors in the aggregate are no long considered accessed. The Bagglink attribute default *accessed anchor (defaccsd)* assigns one of the anchors as accessed when none of the others are.

We use Bagglink in the HyTime/Berlage definition of the CMIF *choice node* in our DTD, as shown in Figure 2. In CMIF, a choice node is positioned at a location in the display. It also specifies several objects, of which one is to be displayed at the node's location at any moment in runtime. Separate CMIF *hyperlinks*, defined with HyTime hyperlinking constructs, associate user-activated objects with the display of each of the choice node's objects.

4.1.3 Sub-schedules as Accessed Anchors

HyTime uses scheduling with accessed anchors to specify that the object positioned in a scheduled event is not one particular object but rather the currently accessed anchor(s) of an aggregate link. HyTime specifies the semantics of an atomic object being contained in an event as an accessed anchor. It allows a coordinate system and its schedules to be contained in an event in this manner, but does not specify how this affects the related scheduling. The Berlage architecture has the *eventfcs* form. It extends a HyTime finite coordinate space by specifying that, when placed in an event through an accessed anchor list, the schedule it contains is positioned within the boundaries of the event within the event's parent schedule.

We use eventfcs to specify the CMIF DTD's *mini-document (minidoc)*, as shown in Figure 2. In CMIF, a mini-document is a component of a choice node that contains the coordinated display of more than one media object. As a Berlage eventfcs form, a Berlage system recognizes that the schedule defined by such a mini-document is to be incorporated into the broader schedule in which the eventfcs is, as an accessed anchor, contained.

5. Conclusion

In this paper we discussed a unified perspective of generic hypermedia structuring and associated presentation specifications as defined by the Dexter/AHM model and by the inter-related ISO standards SGML, DSSSL, and HyTime. The CMIFed hypermedia document authoring and presentation environment was presented in terms of this perspective. The incorporation of these three standards into the design of the CMIFed environment was described.

6. Acknowledgements

Guido van Rossum, Jack Jansen and Sjoerd Mullender designed and implemented the authoring environment CMIFed. W. Eliot Kimber of Passage Systems, Inc., John Buford of the University of Massachusetts and Phil Cooke of Office Workstations Ltd. provided helpful comments on our HyTime usage.

7. References

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

2. C. Goldfarb. *The SGML Handbook*. Oxford University Press. 1991.

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

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

5. International Standards Organization. *Hypermedia/Time-based Structuring Language (HyTime)*. ISO/IEC IS 10744. 1992.

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

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

8. L. Rutledge, J. van Ossenbruggen, and L. Hardman. CMIF HyTime DTD v1.0. <http://www.cwi.nl/~lloyd/Papers/ICCC/CMIF.dtd>, 1997