THE "HyTime"

Hypermedia/Time-based Document Structuring Language

Steven R. Newcomb, Neill A. Kipp, and Victoria T. Newcomb

ILLUSTRATION: SALEM KRIEGER

The computer and telecommunications industries have made enormous progress in communications technology standardization in recent years. One effect of good communications technology is that people can concentrate on the information being communicated. More and more people are realizing, however, that being able to send and receive files containing information is not enough.

It is desirable that all digital documents explicitly indicate in a standard way what kind of notation is used in them. When an electronic document is created, its author should be able to incorporate active references to other on-line documents ("hyperlinks"[1]), regardless of the heterogeneity of their notations. In other words, there should be a standard way to make "information about information" interoperable. Such a standard should, among other things:

- provide a standard way to express the fact that any number of data objects in any number of different notations are related in some way or for some reason.[2]
- provide a standard way to express the way in which any number of data objects in any number of different notations are intended to be rendered for human perception in space, in time, or in both space and time.[3]

One method of incorporating information about the information contained in a document is to use "markup"[4] consisting mainly of "start tags" and "end tags" that respectively precede and follow each logical portion of a document. Tags must be specially punctuated so that the markup can be identified as markup when the document is parsed, and it can be processed separately from the data which it surrounds. Obviously, the rules for specifying the punctuation that distinguishes markup from data must be powerful enough to ensure that the markup and the data cannot be confused even when a given document contains data consisting of examples of markup (for example, an SGML document about SGML).

The proposed standard "HyTime" Hypermedia/Time-based Document Structuring Language (ISO/IEC Draft International Standard (DIS) 10744), built on the Standard Generalized Markup Language (SGML; ISO/IEC International Standards (IS) 8879-1986), is designed to make all this possible. By using SGML/HyTime, all kinds of documents can package their information content using standard "markup." This markup provides information about the structure and notation(s) of the document in a way that is understandable or interpretable by any application that has been provided with an appropriate data importation facility. The "structured" character of such documents will also make them amenable to nonsequential browsing, querying, access and version control, and maintenance over very long time spans.

The World of Structured Documents

The majority of the information stored by human civilization is for perception by human beings.[5] Most of that information is fully formatted for immediate and direct perception, and stored on printed pages. Some of it is fully formatted, but still requires the use of some mechanism to render it perceivable. In this category are such items as magnetically recorded plain ASCII text files, PostScript files, digital video and sound recordings, etc. The rest of the information is stored unformatted, but in such a way that is convenient to format, perform, or otherwise render for direct perception by human beings; databases and structured documents fall into the latter category.

Structured documents are so named because the hierarchical and sequential structure of the various kinds of

information they contain is made explicit by identifying tags. Each tag associates a "generic identifier"[6]—the name of the kind of thing being tagged (e.g., "subsection")—with the data surrounded by a start tag and end tag of the same generic identifier.

Generic tags bear a superficial resemblance to the embedded formatting codes used by virtually all text-processing applications, in that they occur mixed in with data, they can affect the way the data are formatted, and they do not appear in the formatted document. An example of an embedded formatting code is one which causes the text on the following line to be centered, e.g., .ce in an *nroff*[7] document. We might use such an embedded instruction to center the title of an article. However, generic tags and formatting codes are philosophically in different universes. The generic tags in structured documents do not have the effect of associating some particular formatting instruction with some data. That is, in the case of data tagged as a title, no particular formatting instruction is implied; the tags merely identify the title as a title. How a title is to be formatted is entirely a matter to be decided when the document is formatted.

A formatting application for a generically tagged document can be driven by a "style sheet"—a table that associates a set of formatting instructions with each generic identifier. In the case of print documents, title, for example, may be associated with "boldface, 14-point, centered text" and paragraph with "Roman, 10-point, ragged right." If a document is tagged generically, it is generally unnecessary for anyone to edit it in order to reformat it according to a different style. All that is necessary is to use a different style sheet with the formatting application.[8]

Generic tagging offers another even more significant benefit: collections of generically tagged documents can be queried like databases. An example of such a document is: "Show me the authors and titles of the documents whose chapter titles contain both 'napoleon' and either 'frosting' or 'icing.'"

Communities of interest can agree on a set of generic tags for their documents, and they can thereby immunize their documents against the obsolescence of their formatting systems, and at the same time maximize the availability of the information they have collectively created. Moreover, the information in generically tagged documents remains available for uses totally unforeseen when the document was created. A community of interest can further agree about the contexts in which each generis may appear, e.g., "no chapter titles may appear within any paragraph." The agreement can further elaborate each generis with a list of allowable attributes and attribute values—information *about* a paragraph, for example, which is not part of the content of that paragraph.

Why should there be a structuring language for multimedia and hypermedia documents?

People share common languages because they need to communicate with one another. A common structuring language for hypermedia documents is needed in order to permit human communications in this relatively new combination of media, given that the computing environments we create for ourselves are now and will probably always be heterogeneous.

HyTime's design reflects the view that all multimedia and hypertext technologies, all graphics technologies, all audio technologies, etc., regardless of whether or not they are proprietary, should be allowed to compete with one another in a market environment which is technically able to accommodate any combination of them in any information product. It also reflects a conviction that "software" (in the broadest sense that includes such things as movies, recordings, games, databases, news, etc.) is an industry in its own right, very distinct from the hardware and systems industries, and in need of a way to protect its product investments from losing value as a result of changes in hardware technology.

One remarkable feature of HyTime is its ability to accommodate and support musical information in a fashion that will allow such information to be completely integrated with other kinds of information: to share the same scheduling facilities, to be hyperlinkable, etc. Without a deliberate design effort toward a holistic hypermedia structuring language, the representational needs of whole areas of human endeavor, such as music, could be unintentionally prevented from participating fully in our increasingly information-driven civilization. It seems unlikely that a single-industry-driven *de facto* standard would be capable of incorporating facilities that would meet the

[1]The word *hyperlink* is deliberately used in this article instead of the more familiar hypertext jargon *link* because the word *link* has a special meaning in SGML. In SGML, a *link* is a construct used to associate special processing instructions with portions of a document; it has nothing to do with hypertext.

[2]This is accomplished by the document location address module and the hyperlink module of HyTime.

[3]This is accomplished by HyTime's finite coordinate space (FCS) module, including two FCS submodules, the event projection module and the object modification module.

[4]*Markup* is so named because of its resemblance to the markings that editors make on drafts of paper documents.

[5]Comparatively little information is stored in a form intended solely for machine processing, e.g., as a succession of predicate calculus statements, or as an image of a neural network.

[6]The phrase *generic identifier* is SGML jargon. It emphasizes the idea that the identifier is used to say what kind of thing is being identified. It would be just as valid (but not in keeping with tradition) to say *structural identifier*, thus emphasizing the importance of context of the kind of object being identified.

[7]*Nroff* is a venerable text-processing program, usually supplied with Unix-based computer systems.

[8]The SGML "link" feature (which, as noted previously, has nothing whatsoever to do with hyperlinking) does provide an explicit way to associate formatting instructions with the information contained in SGML documents.

needs of as many human communities of endeavor as a committee-developed public standard would. (Although it is often said that a camel is a committee's idea of a horse, there are many people who need camels for situations in which horses cannot function.)

Who is HyTime for?

If there is one industry for which HyTime was created, it is the digital information publishing industry, which now comprises a variety of industries that used to be comparatively unrelated, and which now stand to benefit by participating in each others' markets. By using HyTime, information products can incorporate one another by reference, and many companies can become value-added vendors of the products of other companies. In some sense, the digital information industry takes part in all industry. HyTime can be used as a means of integrating the information management of a diverse enterprise, so that, for example, innovative ideas, their sources, and the obstacles those ideas have to overcome can be identified and tracked, despite the fact that different departments use different machinery and different software. HyTime can be used to facilitate concurrent engineering and other forms of collaborative research, because it can make the documentation and data of all the teams assimilable by all the other teams without requiring anyone to give up his or her existing software investments. Obviously, a HyTime-compliant import/export facility will have to be added to each software system before this can happen.

What HyTime Is Not, and What It Is

Care has been taken in HyTime's design to standardize only those aspects of hyperdocument representation whose standardization will improve the business climate for all competing information vendors. No provision has been made for standardizing user interfaces or user interactions, query languages, or data notations. HyTime does provide standard ways of performing tasks many documents have to do, such as referring to portions of themselves and of other documents. There is nothing to be lost due to the existence of a standard for making such references, and there is much for everyone to gain. HyTime merely standardizes certain rules for creating certain syntactical productions used to express certain kinds of things. The rules are designed to be very general, very extensible, and very unlikely to interfere with competitive product differentiation in any way.

Without HyTime, there is currently no standard way to represent abstract time dependencies. For example, in the case of plans made in connection with large construction projects, progress reports from subcontractors of large projects can always be produced by their own management software, but there is no way to guarantee that such reports can be used as direct input to the general contractor's management software. It is not necessary to reinvent clocks, Gantt charts, management algorithms, or project management software in order to allow the interchange of time dependency data. It is only necessary to agree about how to express an abstract time dependency. HyTime contains a set of agreements about how abstract time dependencies (and other abstract dependencies based on measurement) can be expressed. These agreements were reached by consensus of the industries involved in the standard-making process.

Without HyTime, there is currently no standard way to express the fact that some number of other-

Electronic Review of Documents (ERD)

The paperless flow of documents through enterprises large and small has created version, review, and revision problems for nearly everyone. The problems begin when the authors of electronic documents create multiple versions for different purposes, for example, for email, for typesetting, and for submission to different intramural and extramural publications. When comments come in from several different sources, it can be quite difficult to decide which version the comment refers to, and to manage the database of comments in an orderly way.

The CALS (the Computer-aided Acquisition and Logistic Support initiative of the U.S. DoD) Industry Standards Working Group, which includes parties interested in adding support for the electronic review of SGML text documents to MIL-M-28001A (CALS), has met several times to discuss how such problems might be solved. It has set forth functional requirements [28] for ERD that include the abilities to:

- Associate (hyperlink) comments with text elements at any level in the document structure.
- Provide comment ID information (e.g., review, date, priority, classification/category, status).
- Provide for configuration/version control of comments.
- Provide structure for identification of replacement text, in cases where precise insertion data is known.
- Provide structure to support sorting of comments related to a particular topic or concept ("key" or "topic" attribute, user-defined values, etc.).

The group is pursuing the idea of basing the CALS ERD capabilities on HyTime, because of HyTime's ability to support:

- the creation of specific logical addresses within both text and graphics documents.
- the creation of hyperlinks to document addresses that will work predictably on dissimilar platforms.
- a standard representation for representing the various kinds of activities associated with reviewing documents.

wise unrelated data objects are related or connected in some abstract or exogenous fashion, i.e., no way to express a hyperlink. It is not necessary to reinvent footnotes, hypertext, or multimedia edit decision lists in order to allow the interchange of hyperlinks. It is only necessary to agree about how to express in documents the fact that data objects are related. HyTime contains a set of agreements about how hyperlinks can be expressed. Again, these agreements were reached by consensus of the industries involved in the standard-making process.

HyTime and the Content Data Model (CDM) for Revisable Interactive Electronic Technical Manual (IETM) Databases

The U.S. Department of Defense (DoD) has been working to develop technology and document architecture for revisable databases in support of Interactive Electronic Technical Manuals (IETMs) as replacements for paper technical manuals for logistic support of military equipment [24]. To hold costs down, to immunize IETM databases from obsolescence of presentation and querying systems, and to allow the information to be available on a wide range of different types of machines (from mainframes to CD-ROM belt packs for use in the field), the DoD needs a standard way to represent its technical data. These data include a large number of cross references, access conditions, conditional branchings, possible equipment configurations, etc.; IETMs are hypertexts. With a hypertext standard in hand, the DoD can make compliance with that standard a condition of its documentation contracts with its hardware and software suppliers.

The SGML document type definition of IETM database documents is called the *Content Data Model (CDM)* [15]. The designers of the CDM chose to base its hyperlinking and document addressing facilities on HyTime. HyTime was adopted for a number of reasons, including the following:

- Cross Referencing

  A strong cross-referencing facility was needed. In the course of repairing a piece of equipment, a technician may need to consult a large number of documents, and/or many different parts of each document.
- Conditional Branching

  To identify a technical problem with a piece of equipment, a technician generally performs a number of tests, on the basis of which a diagnosis is made and a repair procedure is prescribed. The manual must outline each step of the testing and repair procedure, and there must be a specific series of steps to be taken for each possible problem. The CDM employs HyTime hyperlinks to represent the possible branchings.
- Multiple Hierarchies

  HyTime can be used to impose an alternative structural hierarchy upon a read-only document, by creating a set of hyperlinks in another document. This feature was regarded as highly desirable due to the large number of anticipated CDM applications uses, and the enormous amount of "legacy data" involved.
- References to Multimedia Objects

  Many IETM documents will have to contain multimedia objects, including motion videos demonstrating inspection and repair procedures, audible spoken instructions, etc. HyTime's standardization of references to digital multimedia objects in all notations will greatly lessen the difficulty and cost of preparing IETM documents for presentation on various platforms and for use in various contexts.

Definitions

The following definitions apply to terms used in this article:

- a *multimedia document* is a parcel of information intended for human perception that uses one or more media in addition to written words and graphics. The presentation of the added media may occupy time, space, or both.
- a *hypertext document* is a parcel of written and graphic information intended for human perception, which can be explored and presented in a variety of sequences, using a set of traversable connections usually called "links" (in this article, "hyperlinks" to avoid confusion).
- a *hypermedia document* is a multimedia document with hyperlinks. The construction and traversal of hyperlinks in hypermedia documents poses special problems, since it should be possible to link to some point with a video object, for example.
- a *time-based document* is a document which specifies one or more time schedules to which its rendering is meant to conform, e.g., musical scores, animation storyboards, plays, and business presentations.
- a *space-based document* is a document which specifies a two- or three-dimensional finite coordinate space, and which specifies the relative positions at which the objects contained in the document are to be rendered. Examples include displays on computer screens comprising blocks of text, graphics, and icons with "screen real estate" allocated to each block—plans for stage sets which allocate space on the stage for each property; etc.
- *HyTime* is a proposed standard

language for representing the structure of multimedia, hypertext, hypermedia, time- and space-based documents.[9] HyTime is a collection of abstract semantic constructs associated with syntactic conventions. It allows hyperdocument interoperability to the maximum extent possible without standardizing multimedia objects, their notations, their modifiers, the effects of those modifiers on them, or the semantics of link types, and without requiring existing documents to be recast in order to make their contents linkable by HyTime documents. HyTime-compliant documents can allow HyTime-cognizant software to browse, render, format, and query them even if that software is not able to understand or render its multimedia objects. If the notation of an object is uninterpretable because no interpreting system is locally available for it, the rendering can still incorporate some form of blankness—darkness, silence, or an appropriate error message—so that the space and time relationships of the rendered and unrenderable objects is preserved.

HyTime Syntax and Semantics

Essential Information vs. Rendering Instructions

When a definition of a document type is created, a distinction is made between information and rendering instructions. This can be a relatively straightforward distinction in the case of many types of traditional print media documents, where all the data consist of printable words and punctuation. In the case of multimedia, hypermedia, and time-based documents such as music, simulations, and process models, the distinction is much less clear. In such documents, the rendering of the information is often critical to its understanding. Only the author or composer can know what rendering information is essential and what should be left as the province of the renderer, or the creator of "style sheet" information. HyTime allows an author to control the rendition by expressing processing instructions as integral parts of the work.[10]

HyTime's "Architectural Forms"

Architecture of SGML Documents

The architecture of an SGML document is expressed in its Document Type Definition (DTD). The syntax is expressed as a set of elements, each with its own generic identifier ("tag"), an optional set of attributes and attribute data types, and a "content model"—a BNF-like production stating what sort of data (or other elements, recursively) can be placed inside each element. For example, a book element may contain several section elements, a section element several paragraph elements, and so on, until the terminal recursive element contains character text. The document architect has complete control over all these features.[11]

Early versions of HyTime

Early versions of HyTime were intended to be included verbatim as an SGML public text entity[12] into another, more application-specific DTD. Each of the early HyTime elements came with its own semantics (documented not only within the DTD, but also in the appropriate ANSI and/or ISO/IEC standard), its own attribute list (which could be added to, but not usually subtracted from), and its own generic identifier (GI). The GI of each element could not be changed—it was the means by which it would be recognized as a HyTime element by a HyTime-compliant application. Ultimately, the committee abandoned the idea that HyTime should be a DTD, because such a scheme had the effect of legislating GIs, thus reducing the expressive power of SGML.

Architectural Forms as Object Classes

Instead of making HyTime a DTD, a more flexible and abstract scheme was developed, in which each HyTime GI became a "HyTime architectural form"—each GI became a fixed value of a **HyTime** attribute. It then became possible to consider each HyTime architectural form an object class, identifiable by the value of its HyTime attribute. By including the HyTime attribute and conforming to the model for a particular HyTime architectural form, document architects can create limitless numbers of subclasses of each such HyTime class.[13] Using the HyTime Architectural Forms, a DTD can be created which incorporates only those semantics of HyTime that are needed (see Figure 1).

HyTime's Modules

HyTime is designed to be used modularly (see Figure 2). Only those portions of HyTime which are appropriate to a given document or application need be supported. A HyTime document may only be a multimedia document, or it may only be a hypertext document, or it may only be a time-

[9]HyTime's representation of time and space is the same for both the time and space measurement domains, and it can be extended to any other measurement domain. Any manifold, combining any measurable domains on any number of axes (e.g., seconds, meters, moles, lumens, grams, dollars) is representable as a HyTime Finite Coordinate Space. Of course, the usefulness of some of the possible combinations is probably greater than that of others.

[10]Rendering instructions are given in HyTime *batons* and *wands*. See subsections: "Event Projection Module" and "Object Modification Module."

[11]It should be noted that SGML only standardizes the metasyntax of an SGML document. It does not standardize the syntax—that is what a DTD does—and it does not provide any way to express or enforce the semantics of an element, even though the document architect may have some very specific semantic in mind. The only way for a document architect to express the semantics of each element and attribute within the DTD itself is to insert appropriately delimited comments in it. Ignored by SGML parsers, such comments indicate the intent of the architect only to human readers, thus helping document authors to use a DTD appropriately.

[12]See the sidebar on SGML for an explanation of the term *SGML entity*.

based or space-based document, or it may be any combination of the foregoing.

The primary modules support basic utility functions (the base module), the indication of segments of information, wherever they may be, (the location address module), the expression of hypermedia links, anchors,[14] and webs[15] (the hyperlink module), and the specification of space and time relationships (the finite coordinate space (FCS) module). Two additional modules can be added to the finite coordinate space module: the event projection module and the object modification module. The latter are used to specify the rendition of HyTime documents. The base module must always be supported by any HyTime-compliant system. In addition, under most circumstances, at least one of the other modules must also be supported in order to take advantage of HyTime's standard semantics.

Base Module

The base module (see Figure 3) in-

[13]The SGML mechanism used to implement the notion of architectural form is the `#FIXED` attribute value data type. When an element is defined in a DTD as a subclass of a given HyTime architectural form, that element definition is provided with an attribute list containing an attribute named `HyTime`, whose value is fixed as the name of the HyTime architectural form to which the element conforms. This has the effect of permanently setting the value of the HyTime attribute to the specified value, in all instances of that element. The document architect must make the attribute list (and, in some cases, the content model) of any HyTime element conform in all ways to the HyTime architectural form named as the fixed value of the HyTime attribute.

[14]*Anchor* is well-established hypertext jargon meaning either (1) a section of data which is designated as an end of some hyperlink, or (2) a section of data which *might* be designated as an end of some hyperlink. In HyTime parlance, *anchor* has only the first definition, and the phrase *document location* is used instead of *anchor* to describe the data indicated by any document location address element, whenever there is no implication that the indicated information constitutes a link end.

[15]*Web* is hypertext jargon meaning "a set of links." It is sometimes useful to create a document containing nothing but hyperlinks. Assuming there is some reason for all the links to occur in the same document, the document might well be termed a *web*.

Standard Generalized Markup Language (SGML; ISO/IEC 8879-1986)

Standard Generalized Markup Language (SGML; ISO/IEC 8879-1986) was developed in response to the need for a way to describe documents in terms of their logical structure. SGML provides:

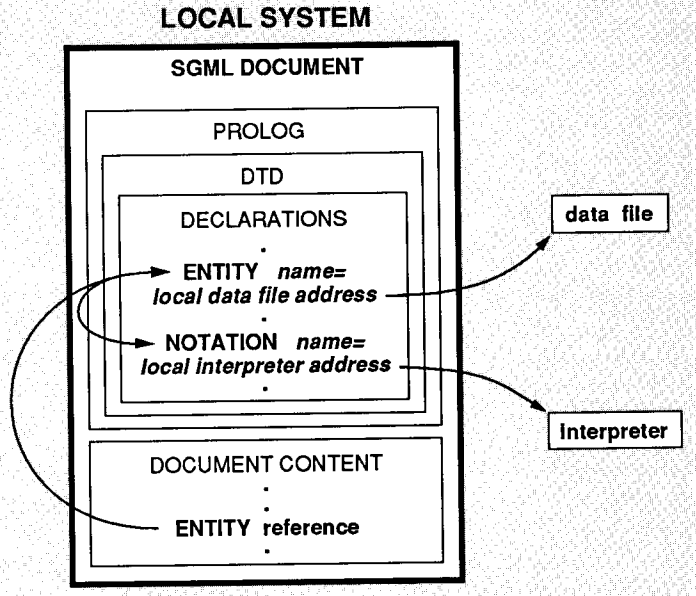
- a metasyntax for expressing agreed-upon syntaxes for individual document types.

Each such syntax is called a "document type definition (DTD)." Each syntax consists of a set of generic codes ("tags") for coding a particular type of document, together with the allowable contents ("content models") and sets of attributes ("attribute lists") for each generis ("element"). Document type definitions have been established by a number of communities, including defense contractors, book publishers, etc. SGML is sometimes described as a "standards machine." There is a set procedure whereby communities of interest can register and publish their DTDs through ISO/IEC standard channels. Such DTDs become "public text entities" able to be included by reference in all document instances intended to conform to them.

- a metasyntax for expressing the syntax of the generic coding ("markup") in the documents themselves (see the subsection "The World of Structured Documents").

While there is a "reference concrete syntax" for SGML-encoded documents which specifies certain character combinations to be used for each markup delimiter (e.g., "</" at the beginning of any end tag), markup delimiters can be substituted, if desired, using an alternative "SGML declaration" at the beginning of a document.

Documents, Entities, and the Local System. SGML is able to allow the same document to appear on dissimilar systems transparently, even when the dissimilarities of the systems require that the document be distributed differently among various files. In an SGML document, an information container (a file) is called an "entity." SGML entity declarations associate names unique within a document with system-dependent information resolvable to the files themselves. (See the figure, "SGML Document on the Local



SGML Document on the Local File System

File System.") SGML also allows references to external programs which process the data in an entity. Declarations of specific data notations are used to associate interpreting systems, for example, a C compiler, or a Digital Video Interactive presentation system, with entities containing information notated in a fashion which may not be parsable as SGML.

Important as the SGML entity management facility might be for ordinary text files, it is even more important in multimedia and hypermedia documents comprising many files in multiple and diverse notations. Porting such documents from one kind of presentation system to another should not compromise their logical integrity, and proper use of SGML can guarantee that integrity.

Why use SGML? There are a number of reasons why various public and private enterprises have turned wholeheartedly to structured document methodology as a general solution for their information-handling problems.

Information whose only format is an explicit expression of its abstract structure is maximally available and recyclable. All SGML documents are amenable to certain very general kinds of processing, including querying and browsing. That means that certain kinds of applications can be used with the entire corpus of SGML documents. Furthermore, it is unnecessary to develop validating software for each DTD. All SGML documents can be validated by a single validating parser which need only determine whether a given document conforms to a given document type definition. Prevalidating documents before formatting them relieves formatting applications of a significant burden of error checking, and many causes of formatter failure can be avoided without bullet-proofing the formatter.

Who Uses SGML?

CALS. Perhaps the single most economically significant application of SGML today is the U.S. Department of Defense's Computer-aided Acquisition and Logistic Support (CALS) initiative. Because of CALS, many defense contractors (including aircraft and ship manufacturers) are required to supply all technical and maintenance information about their products in SGML form. Structured document technology is used to avoid reformatting costs when the time comes to convert its paper maintenance manuals to on-line form, to be delivered as "Interactive Electronic Technical Manuals" to military technicians on "Portable Maintenance Aid" terminals (see the sidebar, "HyTime and the Content Data Model" and [15]). The DoD also plans to use the same technical data in databases so that, for example, a query about the applications of some subassembly, or about the service procedures involving the removal of some part, can be made.

Other Users of SGML. There are many nondefense applications of SGML, including the proceedings of the European Parliament, the *Oxford English Dictionary,* the Text Encoding Initiative (an unprecedented academic effort involving the SGML coding of a stupendous amount of literature), and at least one scholarly multilingual CD-ROM-based bible. SGML is being studied and applied by librarians and scholars serving several scientific disciplines, including various medical fields (e.g., pharmacology, patient records management, biochemistry). SGML is enthusiastically embraced by the air transportation industry. In general, the attractiveness of SGML as a representation tool for a particular body of information increases as that body of information becomes more unwieldy, complex, important, and long-lived.

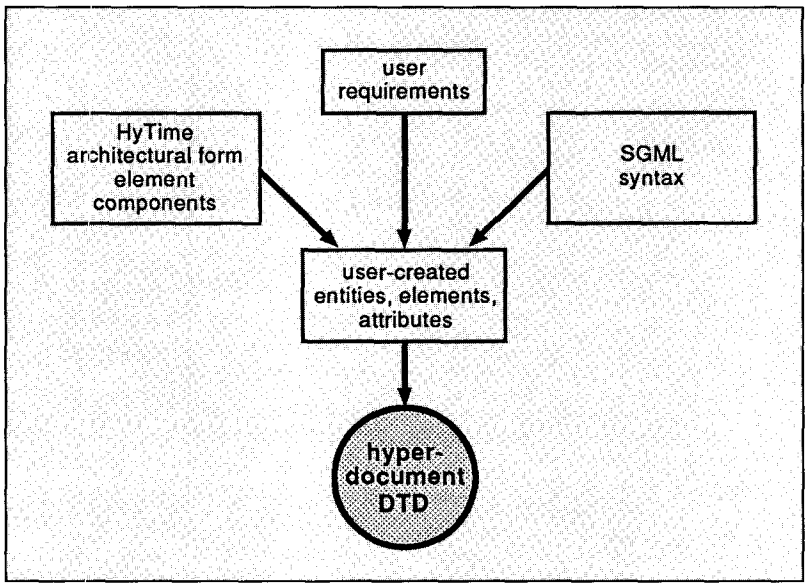
cludes facilities that underlie the facilities of other modules or that are (sometimes optionally) available regardless of which of the other modules are supported. The base module includes:

- *hyperdocument management facilities,* required for all other HyTime facilities: SGML itself, with all its inherent representational and document management conveniences.
- *HyTime identification facilities,* which permit the replacement of HyTime-specific identifiers with user-defined identifiers. These required facilities allow name collisions, however unlikely they may be, to be corrected without compromising the integrity of any document or application.
- *means for specifying application-defined expressions,* called *xenoforms,* in such a way as to identify the notation used. This is a required facility. Xenoforms are combined with other HyTime architectural forms, wherever they were felt to be useful, and/or where there was little point in providing a standard means of expression.
- *coordinate addressing facility* which allows the dimension(s) and positions of events, etc., to be scheduled, and allows document locations to be addressed by position. This facility is required when coordinate location addressing (see subsection "Addressing by Position") and/or the finite coordinate space module (see section "Finite Coordinate Space (FCS) Module") is supported.
- *optional means for specifying activity-tracking policies.* HyTime's location address and hyperlink modules are designed to allow literally anything to become an end of a hyperlink. While it is desirable, necessary, and technically possible to allow such linking, hyperlinks to real-world objects are subject to a kind of entropic degradation, because the linked-to information is subject to change. As a practical matter, it is impos-

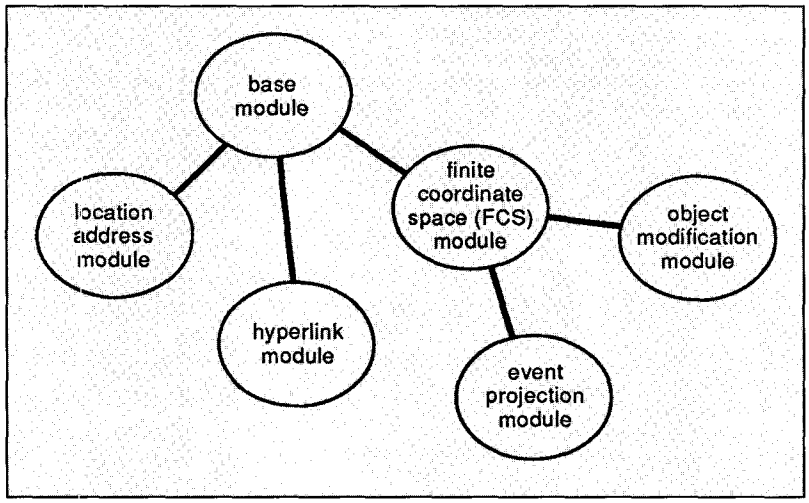
sible either to prevent such degradation or to adapt to it in a timely fashion without the cooperation of the owner and/or editor of each linked-to document. HyTime mitigates this problem (and others) by providing a standard way to establish communications among document owners and editors, and users, creators, anthologizers, etc.

Any element in an SGML document can have an "activity-tracking" attribute, which functions as a pointer to an "activity-tracking policy" element. The activity-tracking policy element can associate policies with any combination of the following activities with respect to the element: (1) creating, (2) modifying, (3) linking, (4) accessing, and (5) deleting. For each activity, there can be a policy expressed in a xenoform.

The activity-reporting facility can be used by the author of a document containing a hyperlink to inform the owner of the linked-to document of the existence of the link, and to request notification in the event that the document (or element, etc.) is revised. It could be used by an appropriately equipped communications system to determine who should be billed and who should receive the royalty in cases where a link is traversed to information for which there is an access charge. The applications of activity reporting in collaborative research endeavors are many. They can be used as a means of collecting opinions, approvals, comments, and links to other relevant material. The access policy can be used for security purposes, to withhold information from unauthorized persons, and/or to report each access. Access reporting can also be used to develop information about the usefulness of the document, and whether it is being used as its author intended.



A HyTime-compliant document type definition (DTD) is created using HyTime architectural forms. Although like any other SGML DTD, it consists of user-defined elements and attribute lists, some of these elements and attribute lists must conform to HyTime architectural forms, which may be combined and/or embellished as needed.



HyTime's modules. Connecting lines slope downward to dependent modules.

- *other optional basic utilities* intended to provide syntactically economical means of declaring default attribute values and definition tables.

Location Address Module

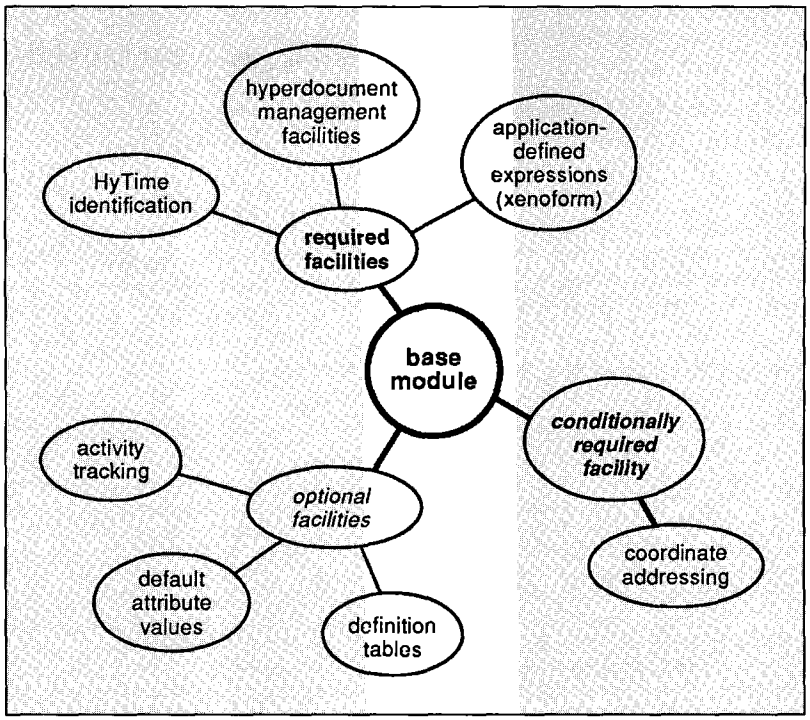
SGML provides a simple way to refer to particular elements in the same document, using its "unique identifier" facility. An identifier (#ID) attribute data type can be used to provide any instance of an element with a name unique to that instance. The uniqueness of the identifier is guaranteed only within

the current local document. Therefore, each document is said to have its own "name space." Addressing is accomplished in this case by using an "identifier reference" (#IDREF) attribute data type, the value of which is the same as the value of the #ID-type attribute that appears on some other element within the same document (i.e., the same name space).

In addition to allowing unique identifiers to be associated with elements, SGML provides a way to name "entities" (files) and to associate convenient identifiers ("entity names") with their addresses on any given computer system or network. These entity names are also guaranteed to be unique within a given document.

Using the SGML #ID-#IDREF mechanism alone, it is possible to represent the addresses of hypertext link endpoints in a standard fashion without the extensions provided by the HyTime location address module.[16] However, the usefulness of the #ID-#IDREF addressing method is limited: only the contents of entire elements within the same local document can be addressed, and then only when those elements are provided with unique identifiers. HyTime extends SGML's reference capability to accommodate those cases in which no unique identifier for an information object exists in the local document's name space. HyTime provides location address (i.e., pointer) architectural forms, which have unique local identifiers, and which contain the information needed to accomplish the task of locating the data.

When a hyperdocument author wishes to refer to an object which does not have a unique identifier in the local name space, that author can create a location address element which pairs the object in question with a unique local identifier. A HyTime-conforming application will recognize the identified element as an address element, and it will know how to use the information it contains to "resolve" the address, i.e., to recover the information pointed at by the address element. Address elements can be chained and aggregated according to the nature and complexity of the addressing job.



HyTime's base module

HyTime supports three kinds of addressing: (1) by name, (2) by position in a coordinate space, and (3) by semantic construct. (See Figure 4.)

Addressing by Name

HyTime's *named location address* architectural form provides a way to address named SGML entities and uniquely identified SGML elements in external SGML documents. It sets forth the information an application will need to locate the file, to establish a parsing context according to the appropriate prolog, and to retrieve the named element or entity.

Addressing by Position

HyTime allows addressing of objects in some arbitrary measurable universe by their position in that universe. HyTime's universes are bounded areas, which are measured in countable atoms called *quanta.* (This is discussed in more detail in "Finite Coordinate Space (FCS) Module.")

All forms of position addressing employ the same "quantum-count" specification scheme used for a variety of purposes throughout the HyTime standard. The dimension(s) and position of the indicated location are specified by counting from the beginning or end of a coordinate axis, or by reference to some other location already specified on that axis. A dimension specification may be defined in terms relative to some other existing dimension specification.

- *String location addresses* allow the specification of substrings. The substring is defined by specifying that the quanta are "bit combinations" (e.g., ASCII characters), and then by giving values resolvable to a starting position and a run length on the parent string.
- *Token location addresses* are like string location addresses, except

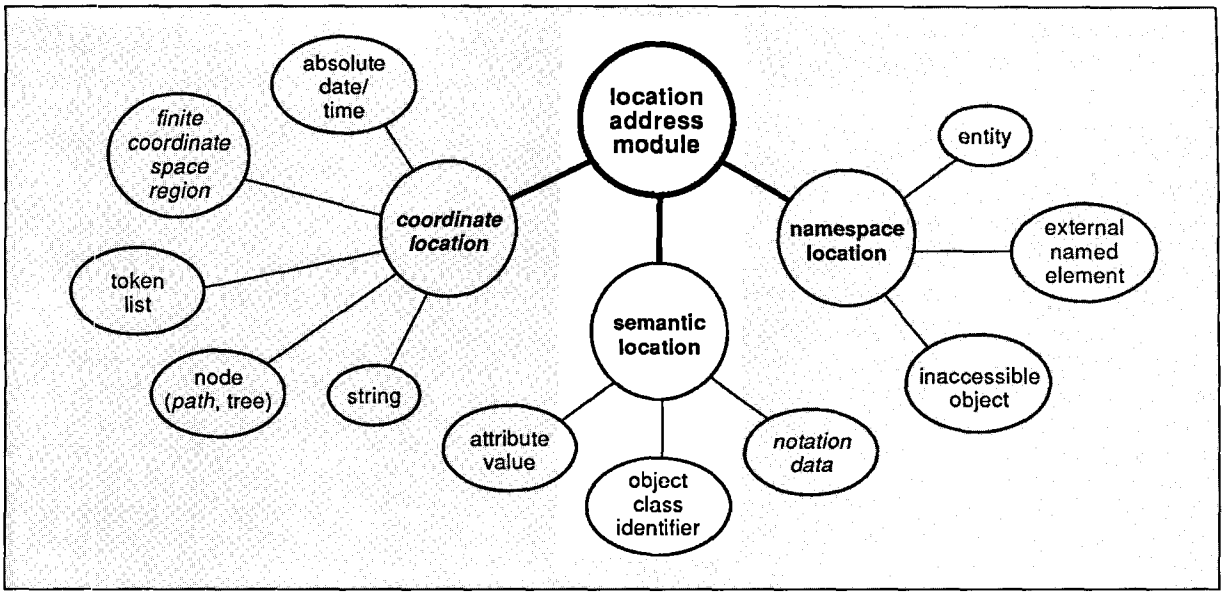


FIGURE 4.
HyTime's location address module. Basically, there are three kinds of addressing: by position (coordinate location), by semantic construct (semantic location), and by name (namespace location). Support of features in italics is optional.

that the quantum is some data combination defined as a token, e.g., a word in a natural language delimited by whitespace.

- A *tree location address* allows a portion of a hierarchy to be identified by specifying target levels by counting levels from the root, and then by specifying siblings, counting on the chosen level(s) from left to right. The quantum is a node.
- *Path location addresses* allow a portion of a hierarchy to be identified by specifying terminal nodes, counting left to right (thus establishing a unique path or paths), and then specifying a range of nodes, counting from the root to the leaf on those unique paths. Again, the quantum is a node.
- *FCS location addresses* (discussed in "Finite Coordinate Space (FCS) Module"), allow the specification of a portion of a media object, such as a photograph, by imposing a Finite Coordinate Space on some actual or hypothetical rendition of the object, and then by identifying the desired portion as a region of that FCS. The FCS location address element provides a way for a portion of the object to be identified when the object's notation is unknown, or when only the rendition, and not the source notation, is available to the author. Obviously, it would often be preferable to use a notation-specific location address, if possible, because greater specificity would usually be possible.

Addressing by Semantic Construct

HyTime provides two architectural forms which allow semantic addressing:

- An *attribute location address* points to the value of an SGML attribute by attribute name; alternatively it may point to a generic identifier.
- *Notation-specific location addresses* allow a subset of the data contained in some larger data object to be identified in a fashion which requires the use of a system capable of interpreting the data. Neither the data nor the expression used to extract the subset of those data need be interpretable as SGML. For example, one can imagine an expression in some application-specific query language for extracting objects from data objects notated in some application-specific page description language, to the following effect: "Please give me only the third polygon from the left."[17]

Hyperlink Module

There are five kinds of hyperlink defined by HyTime, which were designed to meet most general as well as some specialized hyperlinking needs: *independent link (ilink), property link (plink), contextual link (clink), aggregate location link (aglink),* and *spanlink* (see Figure 5). Since each of these is an architectural form, a limitless number of different hyperlink elements, each with its own semantics, may be de-

[16]There are several implementations of SGML-based hypertext systems using the SGML #ID-#IDREF mechanism, and at least some are commercially available.

[17]Notations used to address SGML documents, such as DSSSL addresses or SFQL queries, could also be used as notation-specific location addresses.

fined for any number of applications.

- *Independent links (ilinks),* the most general-purpose hyperlink architectural form, can have any number of link ends. Link ends may point to anchors directly, using the ordinary SGML #ID-#IDREF method, and they may also point to them indirectly, by using the #ID-#IDREF method to point to location-addressing elements, which contain more elaborate addressing information (see previous subsection "Location Address Module"). HyTime also provides a way to specify which link ends are ends from which a traversal can be initiated, and which are ends from which a return can be made. Each link end can be associated with an "end term," which could be some text or an icon to be displayed as a user-selectable "button" which may be "pressed" by a user to initiate traversal of the hyperlink.
- *Property links (plinks)* always have just two link ends, and their purpose is to associate a property (i.e., an attribute name and value) with an element. Plinks are useful when a property cannot be associated with the element in the normal way (e.g., in cases of elements of read-only documents).
- *Contextual links (clinks)* always have two link ends, and one of them is the clink's own location in the document. Clink is the only form of hyperlink whose own location is assigned a specific meaning by HyTime. Clinks could be used in text for footnote references, or, for example, in the context of an FCS event schedule (see "Finite Coordinate Space (FCS) Module") to link an event or event group with something else.
- *Aggregate location links (agglinks)* link multiple locations together in such a way that they are treated as a single aggregate location.
- *Span links (spanlinks)* allow the parsed information contained in contiguous SGML elements to appear to be undivided by any intervening SGML markup. As an example, a spanlink can be used when it is necessary to provide a read-only document with what amounts to markup that conforms to a DTD other than the one originally used.[18]

Finite Coordinate Space (FCS) Module

The finite coordinate space (FCS) module provides for the scheduling of objects. Optional projection and modification modules, for which the FCS module is corequisite, provide mechanisms for controlling or guiding the rendition of scheduled documents (or parts of documents) (see Figures 6 and 7).

Finite Coordinate Space (FCS) Module

In HyTime, an *object* is ultimately any sort of information at all. An object may consist of xenoform notation data; such as digital video and/or audio, a graphic object, a program[19] or script, etc.[20] An object may also be character text data;

[18]Spanlink could be used, for example, to apply versification markup to an SGML-represented Shakespearean play, in which the original markup ignores versification and only identifies the speeches in terms of which actors are supposed to say them. Without spanlink, it would be very inconvenient to mark up verses that run across speaker boundaries.

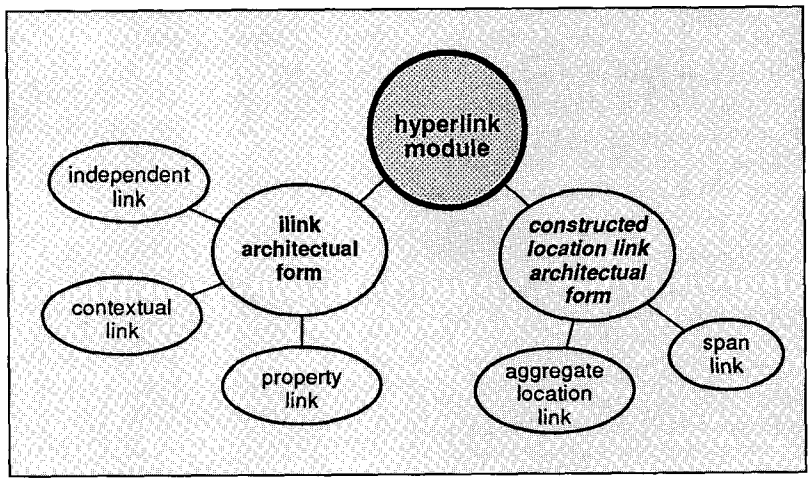


FIGURE 5.
Hyperlink module

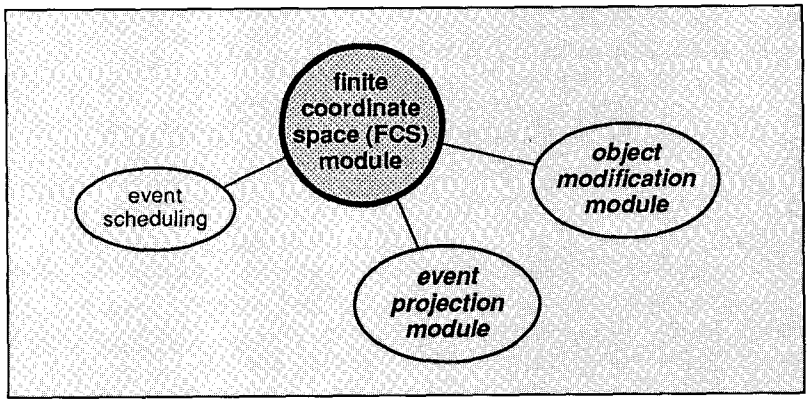


FIGURE 6.
Finite Coordinate Space (FCS) modules. Both event projection and object modification are optional dependent modules.

such data may or may not be parsable as SGML.

Objects in a HyTime finite coordinate space occur in that space as the content of **events**; an event is a nominal conceptual bounding box for an object. Each event has a set of dimension specifications (like those used in location addressing by position) that specify its position and extent on the coordinate axes of the finite coordinate space in which the event schedule (**evsched**) containing the event appears.

In case an object is not readily renderable within the bounding box of its event, HyTime provides a way to express alternative extent reconciliation strategies.

Positions and extents of events in an FCS are specified in terms appropriate to the FCS. Every FCS establishes a specific measurement domain, and a reference unit is defined for each of its axes. All dimension specifications are made using quanta that are fractions or multiples of the reference unit.

For example, a document architect may choose the Système International (SI) meter to be a reference unit, and use millimeters, for dimension specification.

HyTime is provided with many declarations of dimensional quanta based on the SI meter and the SI second. Some prosaic examples include:

- a centimeter is $\frac{1}{100}$ of a meter
- an inch is 2.54 centimeters (an international standard value, not an approximation)
- a minute is 60 seconds
- a millisecond is $\frac{1}{1000}$ of a second

In addition to quanta of real measurement (e.g., second, inch), HyTime allows the declaration of virtual measurement domains with quanta defined in terms of units of *virtual time* and *virtual space*. Virtual measurement units are "figures of merit" in much the same way that dollars are: they mean something with respect to themselves (two dollars are worth twice as much as one), but their rate of exchange to real measurement units, such as loaves of bread, is variable in some application-defined fashion.

There is no constraint upon application developers to use only the measurement granules mentioned in the HyTime standard.

An FCS may contain any number of event schedules (**evscheds**), and each event schedule may contain any number of events. How the events are organized on event schedules, and how the event schedules themselves are organized, is left to the discretion of application designers.

Event Projection Module

The event projection module's facilities are used to specify how the positions and extents of events in one FCS (the "source FCS") are to be mapped onto another (the "target FCS"—see Figure 8). For example, a source FCS of two dimensions might be a large mosaic of square events, each nominally corresponding to a square meter of the state of New Jersey. (For purposes of this example, we will ignore the fact that our planet is spherical.) In the source FCS, each event has a specific position and extent which is described in terms of measurement granules called *meters* from the origin of the FCS. Each event contains an application-defined object which describes that particular plot of land. If we wish to render a map of New Jersey, a simple projector, (e.g., 10,000 "meter" granules in the source FCS to one "centimeter" granule in the target FCS), could have its effect over the entire source FCS, or different **proscopes** could be used to bring different **projectrs**[21] to bear on certain parts of the state, to allow more (or less) detail to be shown for specific areas. The target FCS constitutes a "rendering,"[22] or particular rendered instance of the source FCS—for instance, an exploded view of Atlantic City. A schedule of **proscopes** is called a **baton**. The baton appellation is a metaphor for the baton of an orchestra conductor; an orchestra conductor determines the rate at which virtual units of time (musical beats) are converted into real time. Batons are schedules of proscopes in the same way that evscheds are schedules of events. The content of a proscope is called a projector; the relationship between proscope and projector is similar to the relationship between event and object. As with the objects contained in events, HyTime may or may not understand a given projector. If a projector is in a foreign notation, a HyTime engine may have to ask the application to work out the location and extent of the projected events. However, certain commonplace kinds of projection, such as projection by constant ratio, can be expressed in HyTime.

Object Modification Module

The object modification module provides a way to specify the orderly application of object modifiers (see Figure 8) and combinations of object modifiers. A **wand** is a schedule of **modscopes** in the same way that a **baton** is a schedule of **proscopes**. A modscope expresses the range of the effect of the modifier it contains in terms of the source FCS. Objects or parts of objects in events occupying the same region in the source FCS as that occupied by a given modscope will be affected by the modifier occur-

[19]Such a program could be used to mediate user interaction, for example.

[20]Objects are not always entirely uninterpretable by HyTime. An object can also be a recursive finite coordinate space element, for example.

[21]The missing *o* is not a spelling error. While SGML is capable of handling identifiers of any length, the "Reference Concrete Syntax" defined by ISO 8879-1986 allows a maximum of only eight characters for each identifier. Since the Reference Concrete Syntax constitutes the minimum set of requirements which all SGML systems must meet, all HyTime identifiers have been limited to eight characters for the sake of minimally conforming parsers. However, if conforming to the Reference Concrete Syntax is not required, an alternative concrete syntax can be declared.

[22]Actually, a HyTime engine does not "render" anything. It only creates application-internal data structures which are intended to allow applications to render an FCS in some application-specific fashion.

ring in that modscope. Perhaps the most important difference between event projection and object modification, at least as far as implementers of HyTime are concerned, is that HyTime does far less to standardize the semantics of object modification than it does to standardize the semantics of event projection. Schedules of object modifiers are (somewhat whimsically) called *wands* because, while batons transform scheduled events (constructs fully defined by HyTime), wands act on the objects within events, and neither the semantics of modifiers nor objects are defined by HyTime.[23] The modification of objects has no impact on the position or extent of the events in which they occur.

HyTime Document-Processing Model

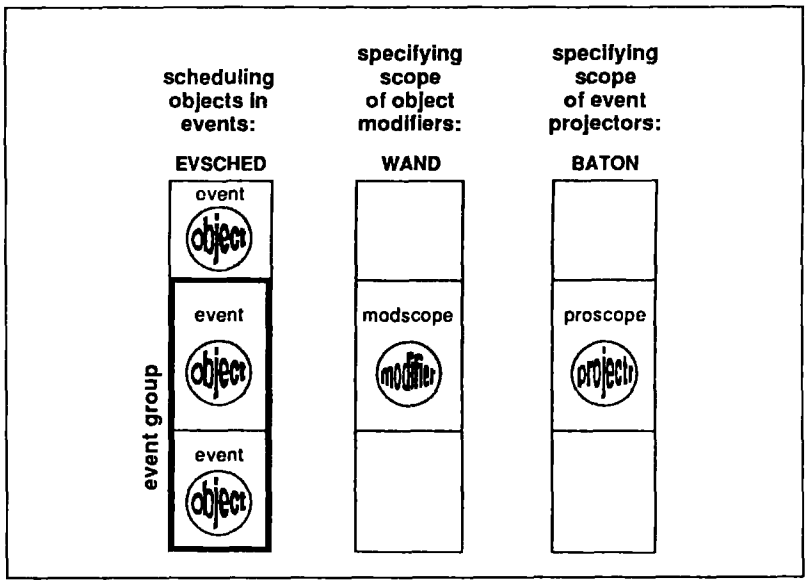
HyTime is designed to help applications developers avoid duplication of effort, and to minimize the difficulty and complexity of making their applications conform to the standard. It is anticipated that "HyTime engines" will be used by systems integrators and applications developers, who will incorporate them into their applications.

The conventional HyTime processing model puts the application in control of everything that happens. When it is time to process the HyTime document, the application calls the HyTime engine, which in turn calls the SGML parser. As it is parsing the document, the parser informs the HyTime engine about everything that it encounters.[24] While this is going on, the HyTime engine does two things:

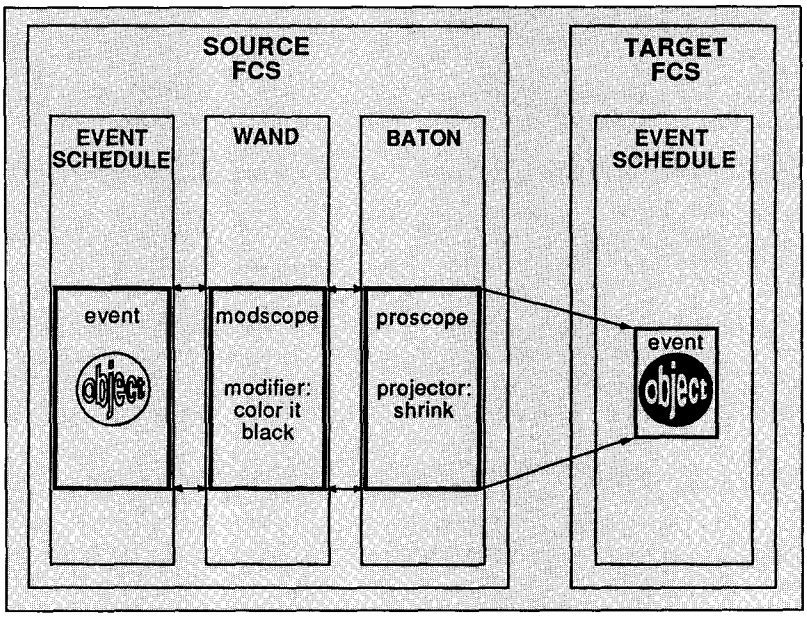
- The engine passes the entire output of the document back to the application. The application uses this information as it sees fit, and it may ignore it entirely.

[23]Therefore, HyTime "understands" batons quite well, but wands are magic.

[24]There is no standard regarding the format of the output of an SGML parser. The nature of that output is an aspect of SGML technology which is left open for vendor competition.



Three kinds of schedules are used in finite coordinate spaces. `Evscheds` contain `events`, which contain `objects`. `Events` define the position and extent of occurrences of `objects`. If the object modification module is supported, `wands` may be used. They contain `modscopes` which in turn contain `modifiers`. The positions and extents of `modscopes` define the areas of the finite coordinate space to be governed by the `modifiers` they contain. Finally, if the event projection module is supported, `batons` may be used; they contain `proscopes` which in turn contain `projectrs`. The positions and extents of `proscopes` define the areas of the finite coordinate space to be governed by the `projectrs` they contain.



Rendering in HyTime: As an event is rendered, the object it contains may be affected by a modifier, and its position and extent may be altered by a projector.

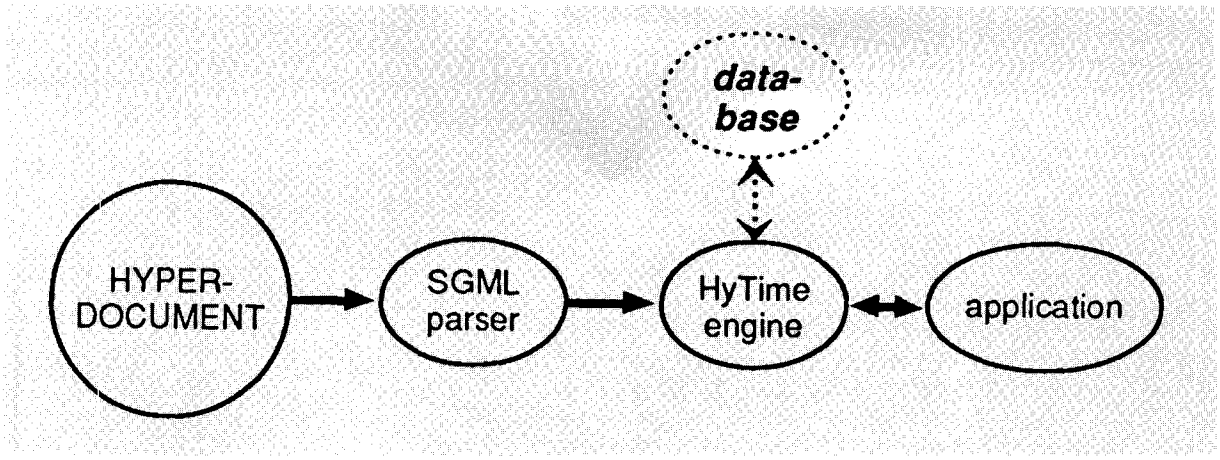


FIGURE B.
HyTime document processing model

- The engine creates an engine-internal data structure based on the HyTime-specific information arriving from the parser.

After the document has been parsed, the application may query the HyTime engine in various ways. The engine assumes responsibility for determining where things are on FCS schedules, for resolving document location elements to the data they indicate, etc.

Although HyTime was developed with the aforementioned processing model in mind, there is certainly no reason why other processing models would not be more useful under some circumstances. HyTime is only a set of mechanisms, not a religion.

Conclusion

To some, it seems incongruous that something as general as HyTime should be derived from an activity intended to result in a Standard Music Description Language. To others it seems natural, necessary, and entirely correct. Musicians have been dealing with the problems inherent in synchronization and real-time rendering of complex documents for a long time. Indeed, Western music notation, as we know it today, has been continuously evolving for over 1,000 years, and it evolved out of at least one predecessor system, Hebrew chant, which is far older. Even (and perhaps especially) today, there is probably nothing more complicated and demanding than producing a live musical comedy with an orchestra, lighting, special effects, and an array of potentially conflicting personalities and artistic agendas. Military people may disagree and claim that producing a musical comedy is almost trivially easy compared with waging a military campaign. NASA people might claim that nothing at all can compare with building a space vehicle and launching it successfully. We live in an extremely complex civilization, and in all three cases (musical comedy, military campaign, and space mission), the problems are actually quite similar, though the stakes differ considerably. They are all concerned with making all the right things happen at all the right times, despite the large number of people and systems involved, and the complexity of the interdependencies involved in the enterprise. Civilization itself can be seen as an invention whose mother is the necessity for predictable outcomes despite unpredictable circumstances. Unpredictable circumstances require contingency plans and the ability to execute the correct course of action when the time comes (i.e., a "late binding" capability).

A fusion of concepts from the world of structured document representation, concepts from the hypermedia world, and certain notions about the specification of rendering intentions borrowed from the music world, HyTime is nothing more or less than a set of expressive mechanisms designed to allow the publishing industry to create and deliver more intricate, more interactive, and more valuable documents while reducing the risk of early obsolescence and loss of investment.

We nourish the hope that, by making information more easily accessible and organizable, and by making it practical for purveyors of multimedia information to address the entire marketplace with a single well-honed and well-maintained document, HyTime will have a salutary effect on the quality of human life. This hope seems realistic, considering the effect that global telecommunications is already having on the global political and economic climate, and considering the magnitude of the improvement in communications that a standard such as HyTime can facilitate.

Acknowledgments

The following persons have each made especially significant contributions to the effort to make SMDL and HyTime a reality: Salim Abi-Ezzi, Larry Austin, Steven V. Bertsche, Carsten Bormann, Garrett Bowles, Rita Brennan, Len

Bullard, Bryan Caporlette, Phil Cooke, William W. Davis Jr., Frank Dwyer, Marion L. Elledge, Edward A. Fox, Anders Franzén, Len Gallagher, Robert Glidden, Ron Gorow, Dorothy Gross, Dave Gunning, Ralph David Hill, Craig Harris, Eitaro Kawaguchi, Yushi Komachi, Francis Kretz, Brian D. Markey, Alan Marsden, Charles Mead, Bernard Mont-Reynaud, Al Nelson, Carlton F. Neville, Tim Oren, Roger Price, Victor Riley, Lloyd Rutledge, Norman Scharpf, Eleanor Selfridge-Field, Donald Sloan, Andy Spicely, Alan D. Talbot, Jack A. Taylor, Hugh A. Tucker, Ludo Van Vooren, Kim Walls, Jerome Wenker, and Wayne T. Wilner. The authors regret that space does not permit recognition of many other individuals who have contributed to the HyTime and SMDL work.

Because of his intellectual leadership, effort, commitment, and sheer tenacity, the contributions of X3V1.8M chair and ISO/IEC DIS 10744 editor, Charles F. Goldfarb (IBM Almaden Research Center) are especially and gratefully acknowledged.

CR Categories and Subject Descriptors: H.1.0 [**Information Systems**]: Models and Principles—*General*; H.3.1 [**Information Systems**]: Content Analysis and Indexing—*Indexing methods*

General Terms: Standardization

Additional Key Words and Phrases: HyTime, SGML

References and Bibliography

Note: Documents described as X3V1.8M/*nn-nn* can be obtained from the Computer Music Association (see Appendix II, final paragraph).

1. ANSI X3V1.8M. Document register. X3V1.8M/SD-2.
2. ANSI X3V1.8M. General information about the X3V1.8M music in information processing standards (MIPS) Committee. X3V1.8M/SD-0.
3. ANSI X3V1.8M. Participants and officers. X3V1.8M/SD-3.
4. Bertsche, S.V. Hypermedia/time-based document (HyTime) and standard music description language (SMDL) user needs and functional specification. X3V1.8M/SD-6.
5. Bullard, L., Price, H., Schoolfield, B., Harlow, R., Domingue, D., Knox, E. and Laffin, M. *Beyond the Book Metaphor: Concepts for Designing and Implementing Integrated Product Development Environments and Digital Technical Information Services.* Obtain from Bruce Schoolfield, GE Aircraft Engines, One Neumann Way MD F126, Cincinnati, OH 45215-6301.
6. Chamberlin, D.D. and Goldfarb, C.F. Graphic applications of the standard generalized markup language (SGML). *Comput. Graph. 11,* 4, 1987.
7. Coombs, J.H., Renear, A.H. and DeRose, S.J. Markup systems and the future of scholarly text processing. *Commun. ACM 30,* 11 (Nov., 1987).
8. *Document Style Semantics and Specification Language (DSSSL).* ISO/IEC Draft International Standard 10179. 1991: International Organization for Standardization and International Electrotechnical Commission, UDC 681.3.06:519.767.
9. Dykiel, R. Technical requirements for ODA/Hypermedia. ESPRIT project. PODA2, Bull S.A. X3V1.8M/91–2.
10. Engelbart, D.C. Knowledge-domain interoperability and an open hyperdocument system. In *CSCW 90 Proceedings of the ACM.* Available from Bootstrap Project, Stanford University. Doc. (AUGMENT, 132082).
11. Goldfarb, C.F. *The SGML Handbook.* Oxford University Press, 1990.
12. Goldfarb, C.F., Newcomb, S.R., Sloan, D. and Talbot, A.D. *ANSI Project X3.542-D, Standard Music Description Language (SMDL).* X3V1.8M/SD-8 (now *ISO/IEC CD 10743*).
13. Goldfarb, C.F., and Newcomb, S.R. *ANSI Project X3.749-D, Hypermedia/Time-based Document Structuring Language (HyTime).* X3V1.8M/SD-7 (now *ISO/IEC DIS 10744*).
14. Graphic Communications Association (GCA). *The SGML Source Guide: systems, software, service, consultants, seminars, and resources.* (A loose-leaf binder publication updated quarterly.)
15. Gunning, D. *Content Data Model (CDM).* Armstrong Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Dayton, Ohio 45433-6503.
16. Halasz, F.G., and Schwartz, M. The Dexter hypertext reference model. X3V1.8M/90–14.
17. Johnson, E. Non-musical applications for the Hypermedia/Time-based structuring language (HyTime). X3V1.8M/90–63.
18. Kipp, N. A document formatter for the standard music description language. X3V1.8M/90–46.
19. Kretz, F. Coded representation of multimedia and hypermedia information. X3V1.8M/90–59.
20. Markey, B.D. Emerging hypermedia standards: Hypermedia marketplace prepares for HyTime and MHEG. *USENIX Summer 1991* (June 10–14, Nashville, Tenn.).
21. Markey, B.D. Multimedia and hypermedia standardization, a report from the ad hoc Study Group on Multimedia Standardization to the ISO/IEC's JTC1 TAG. X3V1.8M/90–73.
22. Newcomb, S.R. Standard music description language. *IEEE Comput. 24,* 8 (Aug. 1991).
23. Oak Ridge National Laboratory. Hypertext bibliography. Apr. 1990. X3V1.8M/90–57.
24. Rainey, S.C., Jorgensen, E.L., and Fuller, J.J. Proposed Draft Military Specification for Revisable Data Base for Support of Interactive Electronic Technical Manuals (IETMs). David Taylor Research Center, Bethesda, MD 20084-5000. DTRC-90/027 (July 1990).
25. Riley, V.A. An interchange format for hypertext systems: The Intermedia Model. X3V1.8M/90–5.
26. Rosenberg, J. Managing time in multimedia. Panel presentation, SIGGRAPH '91, Las Vegas, 91/08/02. (This will presumably be available on videotape or transcribed from ACM SIGGRAPH soon. Call J. Rosenberg at Bellcore, 445 South St., Rm. MRE-2D-292, Box 1910, Morristown, N.J. 07961-1910).
27. Samuel, R.L., III, Ed. User requirements for hypermedia. X3V1.8M/90–44.
28. Stetina, M. CALS Industry Standards Working Group (ISWG) Meeting, 9–11 April 1991. Unisys Corporation, 12010 Sunrise Valley Drive, Reston, VA 22091.

About the Authors:

STEVEN R. NEWCOMB is vice chair of the ANSI X3V1.8M committee (since its inception in 1986), coeditor (with Charles F. Goldfarb) of the HyTime

standard, associate director of the Florida State University Center for Music Research, and president of TechnoTeacher, Inc. He has spoken on HyTime at many SGML-oriented and other conferences, including CD-ROM '90 and SIGGRAPH '91.

NEILL KIPP is a computer scientist at TechnoTeacher, Inc. His master's degree project at the Florida State University Computer Science Department was the first prototype implementation of a Standard Music Description Language parser, formatter, and synthesizer driver on the NeXT platform.

VICTORIA T. NEWCOMB is a technical consultant at TechnoTeacher, Inc. She designs many of the teaching materials used by TechnoTeacher instructors at HyTime tutorial sessions. **Authors' Present Address:** All three authors can be reached at TechnoTeacher, Inc., 1810 High Road, Tallahassee, FL 32303-4408. Internet: {srn,neill,vtn}@cmr.fsu.edu.

APPENDIX I.

HyTime's Provenance

HyTime was originally developed by the American National Standards Institute's (ANSI) X3V1.8M committee, whose original mission was to create a Standard Music Description Language (ANSI Project X3.542-D). X3V1 is the ANSI technical committee responsible for standards in information processing systems for office and publishing purposes. X3V1.8 is the task group of X3V1 concerned with languages for text processing and interchange. Chaired by William W. Davis of Teleprint, Inc., X3V1.8 is the focus of U.S. participation in the ISO/IEC process that has resulted in the SGML International Standard, the HyTime Draft International Standard, and the Standard Music Description Language Committee Draft, among others.

In 1985 X3V1 convened a study group whose purpose was to consider the scope and mission of a proposed Music in Information Processing Standards (MIPS) committee. The study group's report says that, if developed, a Standard Music Description Language (SMDL) should be able to convey both the specific kinds of information that performers generate (play *this* pitch *this* loud at *this* millisecond with *this* timbre, etc.), and the specific kinds of information needed by engravers of common music notation (beam *these* notes together, slur *those* notes together, this is an A-sharp half note, etc.). Thus the study group's report envisions something like a fusion of the kinds of information present in a time-stamped Musical Instrument Digital Interface (MIDI) data stream with the kinds of information present in a music manuscript. A subgroup of X3V1.8, the MIPS committee was dubbed "X3V1.8M" (*M* for music), and it held its first meeting in July, 1986. Since then, X3V1.8M meetings have been held three or four times per calendar year, at various locations in the United States.

In keeping with the U.S. legal requirements which must be met by all standards activities, all X3V1.8M meetings are open to the public—both to those who simply wish to observe and to those who wish to participate. The 68 participants of record in the MIPS committee represent industrial, artistic, military, academic, and personal interests, including large and small business concerns. Software developers, hardware manufacturers, publishers, librarians, composers, musicians, dramatists, actors, dancers, military people, and academic researchers have attended and have been represented at MIPS committee meetings. Several frequently revised "standing documents," including an extensive register of documents contributed to and considered by the committee, a list of participants, an inventory of user requirements, and, of course, the proposed HyTime and SMDL standards themselves, have always been and continue to be available to the public for scrutiny and comment.

Around 1989, it became clear that much of the development work done by the MIPS committee had a large set of natural applications outside the music sphere, and several of those applications, including hypertext and multimedia, were "coming of age" and in need of a set of basic standard representations. Accordingly, the committee was granted an additional project number by ANSI (X3.749-D), and HyTime was created by removing the music-specific mechanisms from the SMDL design, and by making SMDL a specific application of HyTime, which reconstitutes the music language by adding back in the music-specific mechanisms. HyTime has undergone several major revisions and considerable refinement and generalization since its removal from SMDL, but the relationship between HyTime and SMDL remains the same today. A major step in the generalization of HyTime was applying the timing mechanisms to spatial (and other) measurement domains, so as to allow the description, prescription, and rendition of finite coordinate spaces of four (or more) dimensions. HyTime's roots in SMDL are still evident; a schedule of "proscopes" is still called a "baton," for example.

HyTime was "progressed" (as standards jargon puts it) to the Draft International Standard (DIS) phase on August 22, 1991, by a unanimous vote authorized by ISO/IEC JTC1/SC18/WG8. At this writing, the DIS version of HyTime is expected to be published in October 1991. The earliest possible date on which it can be made an International Standard will be at the April 1992 meeting of WG8, which is expected to take place in Denmark. In the meantime, however, as a Draft International Standard, references to HyTime can be included in other ISO standards.

APPENDIX II.

For more information

Special Interest Group on HyTime

The SGML Users' Group Special Interest Group on Hypertext and Multimedia (SGML SIGhyper) maintains a library of documents relevant to HyTime. Printed documents, including copies of the ISO/IEC DIS 10744, examples of the use of HyTime, a directory of parties interested in HyTime, and a newsletter, are all mailed to SGML SIGhyper members automatically as they become available. Annual membership dues vary from $75 to $170, depending on the type of membership and the cost of postage. For more information, contact the authors at the address provided at the end of this article.

In addition, some documents relevant to HyTime are available via anonymous FTP from the University of Oslo, thanks to the efforts of Erik Naggum, the Vice Chairman of SGML SIGhyper. The FTP address in Oslo is "ftp.ifi.uio.no" (129.240.88.1), directory "SIGhyper." All new documents are announced in the USENET group comp.text.sgml as they become available via FTP. An alternative FTP source is at Florida State University in Tallahassee, Florida, "mailer.cc.fsu.edu" (128.186.6.103), directory pub/sgml. Also available at both of the above Internet nodes is the *ARC SGML* public-domain SGML parser, which is written in C, and which was

originally donated through the SGML Users' Group via the University of Exeter.

ANSI X3V1.8M Secretariats

Because of the interdisciplinary nature of the committee's work, there are two X3V1.8M Secretariats. Both Secretariats maintain awareness of the committee's activities among the members of the communities they serve. Both solicit and formulate user requirements, both conduct workshops and meetings, where they solicit feedback on the technical issues confronting the committee, both publish information about the work of the committee in their respective publications, and each has a distinct role in the distribution of committee documents.

The **Graphic Communications Association (GCA)** [14] serves the publishing community's interest in the ISO/IEC DIS 10744 and ANSI X3V1.8M processes by reports in the ⟨*TAG*⟩ newsletter (GCA cosponsors this publication), by hosting committee meetings, and by arranging presentations and discussions relevant to the committee's work at its regular TechDoc, SGML, International Markup, and other conferences and tutorials. In addition, the GCA distributes the committee's documents to the participants of record prior to each X3V1.8M meeting. For more information, contact Marion Elledge, Vice President, Information Technologies, Graphic Communications Association, 100 Daingerfield Road, Alexandria, VA 22314 (tel.: (703) 519-8160; fax: (703) 548-2867).

The **International Computer Music Association (ICMA)** serves the music community's interest in the ISO/IEC DIS 10744, CD 10743, and ANSI X3V1.8M processes by reports in its newsletter, *Array*, by hosting committee meetings, and by arranging presentations and discussions relevant to the committee's work at its annual International Computer Music Conference (ICMC). In addition, the ICMA makes most of the documents listed in the document register available to the public on a cost-per-page basis. Contact The International Computer Music Association, Larry Austin, President, P.O. Box 1634, San Francisco, CA 94101-1634 (tel.: (817) 566-2235; Internet: cma@dept.csci.unt.edu). ■