ELECTRONIC PUBLISHING, VOL. 8(2 & 3), 63–79 (JUNE & SEPTEMBER 1995)

SGML/HyTime repositories and object paradigms

PATRICIA FRANCOIS

Aerospatiale
A/BIS/A - BP D0621
316 Route de Bayonne
31060 Toulouse Cedex 03, FRANCE

PHILIPPE FUTTERSACK, CHRISTOPHE ESPERT

Electricité De France
Direction des Etudes et Recherches
1, Avenue du Général de Gaulle
31062 Toulouse Cedex, FRANCE

e-mail: patricia.francois@avions.aerospatiale.fr, p.futtersack@der.edfgdf.fr, c.espert@der.edfgdf.fr

SUMMARY

This paper proposes a model of an hypermedia repository based on both object technologies (specification method and Database Management System) and electronic document standards (SGML and HyTime).

SGML is an ISO standard for document structuring. It guarantees document exchange capabilities, document longevity and document reusability. Object Database Management Systems fit well for complex document storage and access. However, every time a mapping between SGML and an object model is studied, many issues arise and only a part of them are solved. We propose a complete and generic object model of SGML which eliminates all the limits. Moreover, this model should stand for a universal model, i.e. it can be used as a standard API to plug any SGML visualizer or editor on top of an object repository.

HyTime is also an ISO standard, based on SGML, so that it ensures exchange capabilities, longevity and reusability too. Moreover, HyTime goes beyond the SGML limits concerning the hyperlinking features by offering the semantic to model complex links, such as a link from a document to a very precise location inside another one. We describe how to extend our object model to the hyperlink features of HyTime.

We give an overview of the prototype we implemented to validate our approach.

KEY WORDS SGML HyTime Document management system Object-oriented analysis Object-oriented database management system Hypermedia

1 INTRODUCTION

1.1 A common study between Aerospatiale and Electricité de France

This paper is the result of a current collaboration between Aerospatiale Aircraft Business and the research and development division of Electricité De France. This collaboration concerns a study and research project in the electronic documentation storage and management field. Although the specific industrial contexts are different, numerous common requirements may be identified in this particular field and a large benefit may be expected from a common study.

After presenting this study's industrial contexts, we identify in Section 2 our common needs. Then, in Section 3 we describe the SGML/HyTime repository architecture on which our study is based. In Section 4, we show how we have chosen to implement this repository. Finally, we conclude by giving the progress status of our work and the main issues which remain to be studied in depth.

CCC 0894–3982/95/020063–17
©1995 by John Wiley & Sons, Ltd.

Received 1 April 1996
Revised 5 July 1996

Aerospatiale and Electricité De France are two big French companies which produce respectively, aircraft (Aerospatiale Aircraft Business) and electricity. Both need to manage a large amount of documentation in their own industrial context. As a consequence, a significant benefit is expected from powerfully computerized documents.

1.2 Aerospatiale industrial context

Aerospatiale Aircraft Business is responsible for producing all the technical documentation delivered with aircraft. And this aircraft technical documentation is subjected to severe constraints, particularly in terms of volume (more than 300 000 pages for one kind of plane), content format (textual, technical data, illustrations), content customizing (airline customizing), authoring (performed by various industrial partners), update frequency and longevity requirements.

Since the 80's, paper media has been giving way to electronic media in the aerospace community. And the SGML format has been adopted by the aerospace regulatory organization (ATA — Air Transport Association of America) as the documentation exchange standard between aircraft manufacturers and airlines. As far as SGML structured documentation is concerned, Aerospatiale Aircraft Business is involved in three main domains: standardization, documentation production, documentation utilization software development.

Aerospatiale participates in various military and civil standardization committee groups which design DTDs for aircraft documentation delivered to airlines. In terms of documentation production, SGML technical publications are available, since 1993, for new aircraft: Airbus A330/A340. Moreover, a large documentation system re-engineering project is going on; it aims at integrating new documentation technologies in the documentary product as well as in the production process. In terms of documentation utilization, Aerospatiale proposes a software, called ELS — Electronic Library System, which allows aircraft documentation delivered to airlines to be integrated into their own information system.

Aerospatiale Aircraft Branch is also involved in research and prospective activities in the electronic documentation field. These activities mainly consist in studying new standards and technologies related to electronic documentation, in order to evaluate their adequation to aerospace needs and specificities, i.e. their applicability in the aerospace field. Such research activity development aims at preparing future aircraft documentation and related technological evolutions in the three above-mentioned domains: standardization, documentation production and documentation utilization software development. One of these research activities concerns the storage and management of hypermedia aircraft documentation.

1.3 Electricité De France industrial context

Concerning Electricité De France, the main goal of the R&D division is to do research on electricity topics. From the information system viewpoint, the research activity tends to generate technical documents. But, as a complement to electricity research, we assume an activity on scientific topics such as computer science and documentation engineering in particular. In this context, we are studying a system to manage electronic documents. The Electronic Library Project aims at studying the documents concerning general activity of the division.

In the context of general activity, we keep track of the division flowchart, of the people employed by the division, and of accounting information. Moreover, the employees produce sets of electronic documents to describe what they intend to do, and later, the corresponding reports. As examples, we can mention activity descriptions (about 2000 documents of 2 pages each year), activity reports (twice a year) and internal technical reports describing general results of research actions (5000 documents of 10–100 pages each year). All these documents are generally written with a very popular word processing tool.

This information is collected through the office automation network and stored in relational databases. Only the title, the abstract, and the authors of the internal technical reports are stored in a coded format. Internal technical report bodies are digitalized (because of the heterogeneity of the collected documents) and stored by a specific application on optical disks.

Many applications manipulate these data, to print or fax a report, to retrieve a selected set of information, or to compute synthetic results by using natural language and statistical techniques.

2 CLASSIFICATION OF NEEDS

Hereafter, we present first the technological needs, then the applicative (functional) ones and last the consequences for the repository model described in this paper.

2.1 Needs in terms of technologies

The main need is to work on structured documents. We chose the universal ISO format: SGML [1], mainly for exchange, longevity and reuse reasons [2,3]. This format offers all the necessary features for managing structured textual documents possibly linked with non-textual documents. SGML also offers mechanisms to represent simple links and shared sub-documents. However, those features are not powerful enough to satisfy our hypermedia needs. So we will introduce the more general features of HyTime in Section 3.3 below.

The second need concerns document storage. We have to maintain reliability of a huge amount of (sometime big) documents which are accessed by many users. So, our documents must be managed by a DBMS. Moreover, these data have the particularity of being complex as opposed to flat data. Indeed, we store documents split into numerous objects corresponding to each of the SGML concepts (Element, Entity, Entity reference, Attribute, Element declaration, Attribute declaration, etc.). All these data represent granules of information strongly interconnected one to another. The application needs (Section 2.2) also involve complex data processing. For modularity reasons, those treatments must be broken in many small methods attached to small pieces of information. So we use object technologies by basing our projects on an Object-Oriented Analysis (OOA) specification method [4] and an Object Database Management System (ODBMS). As a consequence, we benefit from the direct mapping between the recursive tree-structured SGML documents, their object specification and the database structure which receives them. The adequacy of ODBMS to SGML document storage has already been demonstrated in [5,6,7,8]. Here, we just extended the principle by introducing an OOA specification as an intermediate layer. It enables to define an abstract formalism independent from the ODBMS and structured document models. This formalism is used to describe our repository model.

The reader will note that the two representations are complementary and not redundant:

on one hand, SGML and HyTime are the ISO standards which guarantee the data longevity and exchangeability. On the other hand, the object representation enables the processing of the data in a database for current information management.

2.2 Needs in terms of applications

In this section, we describe first the collaborative editing issue, second document composition, third versioning/variants features, fourth virtual document extraction, and last queries and navigation facilities. These five issues are ordered from the most sophisticated kind of application to straightforward features supplied by the ODBMSs. They are further developed in [9,3].

2.2.1 Collaborative editing

In our context, the “SGML repository concept” arises from a base requirement: the collaborative editing of SGML documents. The SGML repository is designed to be the core of an editorial workbench where a group of people collaborate in the manipulation of SGML documents. Because of the large size of documents processed, people often access concurrently various parts of a whole document. The issue consists of locking a fragment of information, corresponding to a subtree of a structured document. This part of the document can then be modified by a single user according to the document model. Then, an unlock operation must be done to release the piece of information for the other users (other editors).

There are two ways to perform collaborative editing. The current one consists of extracting the subtree of the document from the database and browsing it to the end-user through an SGML editor like in [10,11]. The second way, which is still in the research domain, consists in authoring directly in the database. In this case, no fragment is extracted from the database; the editor works directly on the data in the database. It seems to be the best solution because SGML is contextual, so that it is not possible to extract a very limited subtree without losing a part of the context.

2.2.2 Document composition

Document publication is the final step before giving the information to the reader. Indeed, it is important to make different presentations according to reader profiles and the media (paper, screen, etc.) on which the document is displayed.

2.2.3 Versioning and variants

A history of the content and/or structure modifications must be kept. It is also important to manage variants of documents, according to the languages (or the aircraft configuration for an aircraft technical publication for instance).

2.2.4 Virtual document extraction

The virtual document is a document generated by a query on demand. It is necessary to compose a document according to specific uses. For example, composition of a French document from a multi-lingual one or composition of a maintenance manual for an aircraft with a Rolls-Royce engine, by extracting adequate subtrees (variants) from the source document.

2.2.5 Document conversion

Another goal is to build new documents by assembling several or parts of several documents, or merely to modify documents directly in the database.

2.2.6 Queries and navigation

Queries are used as filters on the structure and/or the content. For users who prefer to explore the database without data structure knowledge, hypertext navigation is more adapted.

2.3 Induced needs in term of repository model

According to the needs described above, we may identify the following three major constraints.

2.3.1 A generic model

Our goal consists in defining a generic model, that means a completely DTD-independent object model based faithfully on the SGML meta-model. In this way, we will obtain a very powerful model capable of managing any type of documents. As this generic model is predefined, we avoid any model modification and any extra-development for supporting a new document type. It is a major advantage in our context because we need to manage numerous DTDs which may frequently evolve. This generic approach differs from the more specific approaches developed in [12,13,14] which propose partially DTD-dependent models. These specific approaches are well suited for enabling direct usage of the native ODBMS query language as explained in [13] or performance enhancements like in [12]. However, they induce meta-schema manipulations and so would introduce limits in our context.

2.3.2 A complete SGML/HyTime model (Base Constraint)

Within an editorial workbench, two levels of data modeling can be identified: the exchange model (SGML) and the repository data model (object model). So it is necessary to ensure that no information is lost when documents are transformed from one model to the other.

One of the SGML editorial workbench specificities is that the SGML ISO standard is used both for exchange between the repository and the tools around it as well as between the workbench and the rest of the world. To perform all these exchanges (fragment extraction, re-integration after use) without losing information, implies that the data model within the repository be “equivalent” to the exchange model itself, i.e. it takes into account all the SGML constructs. Another specificity of the SGML standard is that this format is used as a rich structuring model for documents. To take full advantage of an SGML document within the repository, therefore, implies that the data model in the repository be at least as rich as the exchange model, i.e. the SGML structure.

In actual fact, these two models are closely related and constrained to evolve together in a consistent way. So when designing the repository data model, we will always have to aim at satisfying the above-two constraints which are for us the major and base ones. In our approach, we therefore pay particular attention to obtaining a complete object model of the exchange model concepts (whether SGML or HyTime). As we often refer to this constraint in the paper, we call it “the Base Constraint” [9].

2.3.3 Universal object model

As a consequence, we aim at developing a universal object model, consisting in an interface which enables any SGML/HyTime native tools to be plugged in the SGML/HyTime repository.

3 THE SGML/HYTIME REPOSITORY DATA MODEL

3.1 Our approach characteristics

3.1.1 A strategy based on the OOA method

The OOA method is based on a strategy for rigorously organizing a software development process. This strategy is based on the concepts of Domain and Subsystem [4]. A domain is a separate and independent set of objects that behave according to rules characteristic of the domain. A large domain is partitioned into several subsystems on the base of the following rule given in [4]: “*objects on an information system tend to fall into clusters, group of objects that are interconnected with one another by many relationships. By contrast, relatively few relationships connect objects in different clusters*”. When partitioning a domain, we divide the information model so that the clusters remain intact. In this paper, we will use the word “class” rather than “object” (OOA vocabulary) or “entity” (entity/relationship model vocabulary) which may be ambiguous (confusion with SGML entity . . .).

3.1.2 A step by step approach

We have chosen to take full advantage of the OOA partitioning strategy for designing a repository data model which satisfies all the requirements identified. We have then followed a step by step approach for analyzing the application domain.

First, we have identified the SGML Document Instance Subsystem and we have defined a base information model able to meet base requirements. Then we have kept on analyzing our domain for satisfying all the other needs step by step. This approach led us either to enhance that base model, or to define other subsystems, depending on how much newly defined objects are interconnected with previously defined ones.

Finally, we obtained the following architecture shown here (cf. Figure 1); it will be detailed later on in the following sections.

3.2 The SGML repository data model

3.2.1 The base model

Our strategy for defining an SGML document base model has been described and analyzed comparatively to related work [5,6,8] in [9,2,3]. We just sum it up here; we will describe

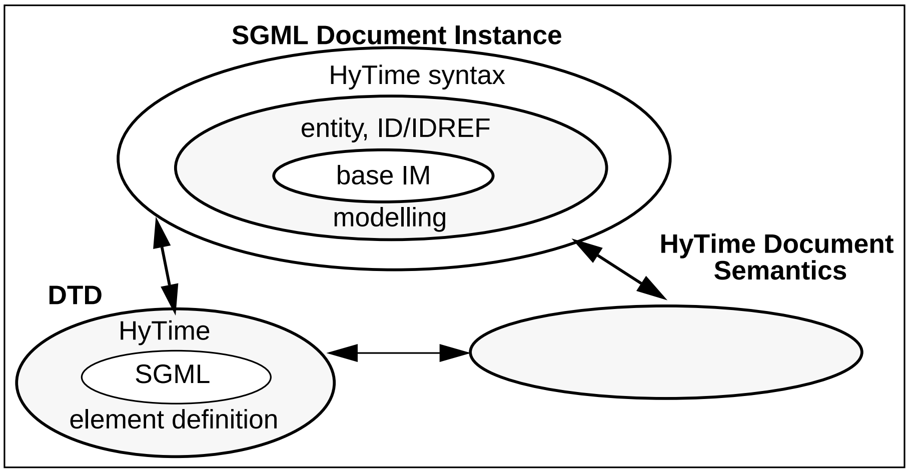

Figure 1. SGML/HyTime repository application domain and its subsystems

in this paper our strategy for evolving towards a hypermedia repository. Unlike [5,6], we propose a fully generic base model that means a completely DTD-independent model. And, in order to satisfy the Base Constraint, this base model is derived from the SGML abstract syntax. Like the ESIS mechanism, specified in an SGML annex [1], which defines a set of information on the element structure, we used the same philosophy to get information from the complete SGML abstract syntax. The ESIS consists of a flow of information generated as the document is being parsed; it is defined to be the relevant set of information for recreating the source document as well as for implementing any structure-based application. Figure 2 shows a simplified subset of this base model. It corresponds to a tree of SGML components decorated with SGML attributes.

3.2.2 Enhancements for satisfying other requirements

Fine DTD modeling is required for authoring and to perform automatic conversions. Indeed, the analysis of requirements revealed a need in terms of document conversions, whether manual or algorithmic, within the database. Performing authoring or launching conversion scripts directly within the database both require performing document incremental parsing within the database. Each time a primitive transformation is performed (e.g., inserting an SGML element or updating an attribute value etc.) it is necessary to check whether this transformation is allowed according to the DTD rules. We therefore require an efficient access to DTD rules that means a fine DTD modeling. This DTD model is presented in [9,2].

In fact, the classes related to the DTD fall into one specific cluster and have few relationships with the classes related to the SGML Document Instance. So we have then identified a new OOA subsystem for DTDs (cf. Figure 1). This subsystem may be ignored by any application which is not interested in database direct transformations.

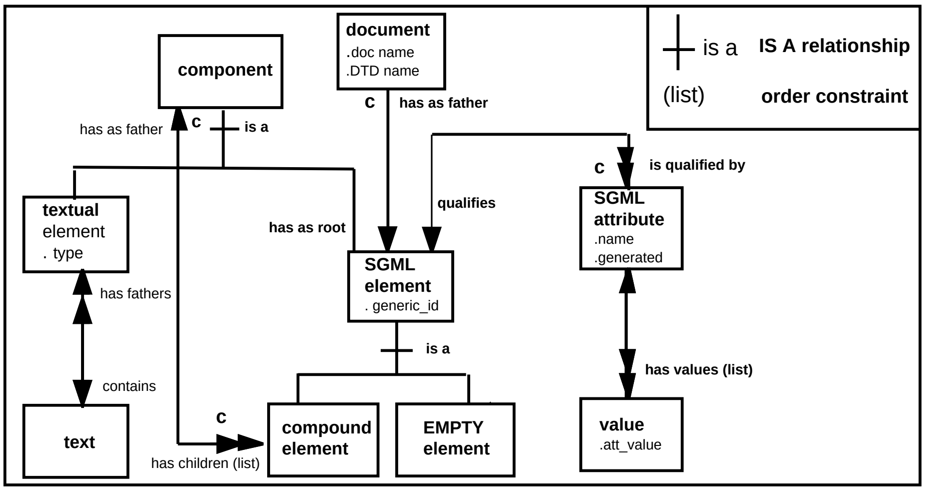

Figure 2. The SGML Document Base Model

3.3 Towards a hypermedia repository data model

3.3.1 Our approach characteristics

Whether in a technical publication production or utilization context, the document concept is greatly evolving. On one hand, documents become largely inter-dependent; they reference each other, share components and so forth. On the other hand, documents tend to only appear as end-products that result from an assembly of documentary data. Managing a web of documentary data connected by various kinds of links, rather than a collection of independent textual documents is therefore an increasingly emerging requirement. The repository tree model has to evolve towards a graph model. This graph model is close to the one defined for hypertext documents [15] in that they both allow non-linear and interactive functionalities to be provided. But unlike most of hypertext-related work [16,15,17], we are constrained to deal with standards for exchange, longevity, and regulation reasons. Moreover, most of this research work focuses on the global graph structure definition. Here, we need to manage a graph whose nodes have their own SGML complex tree structure (as in [18]).

The SGML exchange model offers limited hypermedia features which are not sufficient for satisfying all our requirements. So we have chosen to evolve towards the HyTime [19] exchange model which is an ISO exchange standard, fully up-compatible with the SGML standard. Based on the SGML syntax, HyTime introduces a new concept: the Architectural Form concept. An Architectural Form is a set of rules whose semantics are used to specify DTDs. Architectural Forms allow hypermedia features to be modeled (hyperlinks etc.) as well as time-based features (scheduling etc.). A HyTime document therefore consists of an SGML document, some elements of which have standardized semantics.

The SGML repository model defined above only represents the SGML tree structuring constructs which may be qualified as syntactical constructs. This model consists of a direct mapping of the SGML tree model; exchange and repository models are then isomorphic. As a consequence, for each SGML document to be imported within the repository, this model may be instantiated as the document is being parsed e.g. sequentially read. In the same way, preorder traversal of the tree model allows any marked-up source document to be recreated.

But, as far as hypermedia (whether in SGML or HyTime standard) features are concerned, semantic concepts are added to some syntactical constructs. And these semantic concepts have to be largely known and used within the repository for providing rich hypermedia capabilities (browsing, etc.). The repository data model therefore not only consists of a mapping of the syntactical constructs but has also to be semantic-aware.

Note that HTML is not in opposition with our SGML/HyTime strategy as HTML is nothing but an SGML DTD. However, as described in [9], HTML can not fulfil our requirement because:

- it is restricted to only one DTD,
- it does not enable to address segments with a sufficient granularity,
- it does not offer semantic on the links,
- does enable to keep link validity if referenced documents are modified.

For the reader who knows the hypermedia Dexter model, the HyTime concepts are close to the Dexter extended model presented in [17]. As demonstrated in [20], one of the main differences is the presentation independence of the Hytime approach. At the opposite, the Dexter approach embedded the presentation inside the hypertext structure. Moreover, the Dexter “within component layer” can be compared to the document format independence of HyTime. So we study here the SGML/HyTime hypermedia approach.

Our strategy for evolving towards a hypermedia repository model therefore consists in:

- Partitioning the repository model into a syntactical and a semantic layer, taking full advantage of the OOA partitioning strategy. The syntactical layer consists of a mapping of the exchange format syntax. It ensures equivalence between repository and exchange models (cf. Base Constraint). This syntactical layer may be instantiated as the source document is being parsed. And it is designed to be the only one to be invoked for exchanging any hypermedia document. The semantic layer represents document hypermedia semantics.It enables hypermedia rich capabilities to be provided by the repository. This layer is instantiated in a second step and results from a calculation performed by querying the syntactical layer. The semantic layer is then the result of what we will call “HyTime processing”. This semantic layer only belongs to the repository and never needs to be invoked when recreating a source hypermedia document, for exchange purposes. Following our general goal, we aim at defining a generic semantic layer based on the HyTime semantic model itself; this in order to ensure that no extra-development is necessary for supporting any new hypermedia document type (in opposition to [5] where a specific semantic layer model is proposed for optimization purposes).
- Defining the repository hypermedia model in two steps: the SGML and the HyTime hypermedia features. First, the existing SGML hypermedia features (entities, ID/IDREF attributes) are modeled because they consist of base features used by HyTime. This strategy enables an evolution from SGML to HyTime to be performed

in both a smooth and consistent way as described in Section 2.3.2. Second, a part of the HyTime hypermedia features [21] are modeled as detailed in Section 3.3.3.
- Validating this modeling work in an implementation phase based on the first HyTime applications: the Grund DTD [22], and CApH [23].

3.3.2 *First step: taking into account SGML hypermedia features*

Entity construct modeling is required for information sharing. Whether in a document production or utilization context, managing information sharing is a major requirement. A mechanism which ensures that exactly the same information component is included or referenced in various document fragments must be provided.

The SGML standard provides a virtual storage concept through the entity construct. An entity may contain SGML or non-SGML data, possibly referenced from different documents. These anchoring or referencing mechanisms finally consist of special links between documents (like the ones called “alive copy” links in [10]). A first step for evolving towards a hypermedia repository consists in managing this construct within the repository, i.e.:

- enhancing the base model for finely modeling the entity construct: this is mainly for ensuring equivalence between exchange and repository models,
- taking into account the ENTITY semantic concepts within the repository: during document consultation, for instance, information sharing has to be transparent for the end user. Components shared have to be displayed when required. But when updating data within the repository (update, insertion, deletion, etc.), consistency of data and particularly of shared components has to be controlled and managed. Entities references therefore have to be interpreted as particular links, “anchoring links”, within the repository (for browsing capabilities) and managed as “logical sharing mechanisms” (consistency control).

Finally, the repository base model has to be enhanced (cf. Figure 1) for modeling both entity syntax (syntactical layer) and entity semantics (semantic layer). But due to the ENTITY construct complexity, these two layers are mixed and not easy to differentiate within the resulting model (cf. Figure 3). However, we may identify the semantic relationship which represents an anchoring link.

ID/IDREF attribute modeling satisfies the document traversal link requirement. SGML provides a standardized way of modeling cross-references within a document; this is by means of a specific kinds of attribute: ID/IDREF attributes. Enhancing the base model (cf. Figure 1) to take into account these cross-references would enable traversal links to be managed within the repository (browsing capabilities).These enhancements (cf. Figure 4) consist in:

- extending the SGML attribute model for identifying IDREF attributes (syntactical layer).
- adding real traversal links within the repository (semantic layer).
  Instances of these traversal links are calculated by searching the SGML component identified by the IDREF value, e.g. by querying the syntactical layer. These instances are useful for navigating through cross-references but their integrity has to be checked when documents are updated.

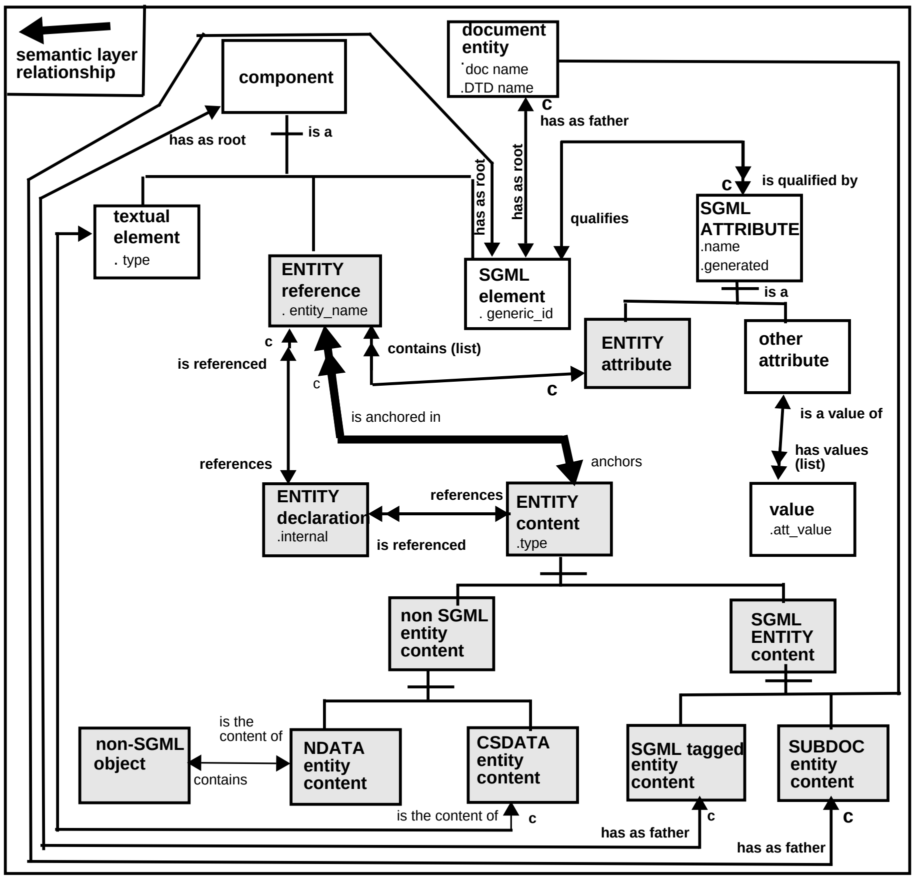

Figure 3. Base model enhancements for entity modeling

Finally, this first step identifies a major characteristic of the semantic layer. The semantic layer is not only affected when documentary data are imported within the repository. Each time documentary data are modified (documentary data updates, insertion or deletions), some related semantic data may become inconsistent. Properly management of semantic concepts (such as information sharing, cross-references) within the database therefore requires implementation of complex integrity checking mechanisms on the semantic layer, in order to generate consistent and automatic update of semantic data each time their related syntactical data are modified.

3.3.3 Second step: HyTime hyperlink features modeling

- Requirements

Many kinds of link can appear in a document:

- references to textual pieces of information contained in other documents,

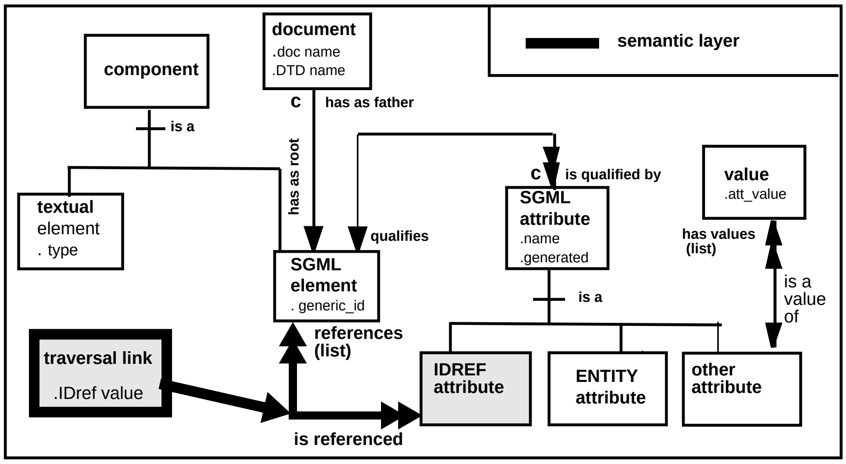

Figure 4. Enhancements for traversal link modeling

- references to non textual objects (graphics, etc.) Locating information included in a non-SGML object implies the use of a language that can be understood by this object (e.g. locating a portion of a graphic).
- references to remote technical databases. In the same way, the locating mechanism associated with these references must be capable of being understood by the external database (e.g. SQL).
- complex typed links between many documents. This is to enable pre-selection according to user profiles for instance like in [24].

Consequently the exchange format has to propose a standardized way of modeling typed links combined with powerful location mechanisms.

The SGML entities enable modularity and information sharing to be managed. However, they only allow the anchoring of a whole module, whether SGML or non-SGML, within a document; they do not propose a mechanism for locating information in an entity. The SGML standard provides a mechanism for modeling intra-document links but not typed inter-document links.

Conversely, the HyTime standard proposes a standardized way of modeling typed links combined with location mechanisms which are partially based on the Entity and ID/IDREF constructs [25,19]. Evolving towards a HyTime repository will then allow all the above requirements to be fulfilled.

- Approach

In order to satisfy the Base Constraint, we have defined a base model derived from the ESIS definition, specified in the SGML standard first versions [26,1]. But this logical flat representation of an SGML document (which may be easily mapped into an abstract tree) turns out to be limited particularly for representing hypermedia documents or for specifying complex document transformations. The HyTime and DSSSL working groups [27,19] have then worked together to enhance the concept of property defined in HyTime. The HyTime standard corrigendum defines an exhaustive property set representing all the information a parser is capable of making available about a document. Graphs, where arcs are defined in terms of properties, may be built. Such graphs are called "groves". But, this standard also provides a way (by means of a grove plan) for any application to get a grove that provides it only with the information it requires.

In order to continue to fulfil our major constraints (cf. Section 2.3), we aim at defining a HyTime repository model compliant with these grove and property set definitions. In a first phase, we cannot reach a complete compliance but we can rigorously specify our compliance level by defining our application grove plan. We will then enhance it step by step.

- The HyTime repository model: the syntactical layer

The HyTime standard is based on the specification of architectural forms (cf. Section 3.3.1) which are grouped into modules. The base module is required as it specifies base architectural forms. The other modules are optional. In a first phase, we will only focus on two of them: the hyperlink and the location modules. The hyperlink module specifies architectural forms for representing links between any kind of documentary data. These link end-points are located by means of location mechanisms specified within the location address module.

All the architectural forms defined in the standard constitute the HyTime meta-DTD. Then any of these architectural forms can be used or not in any DTD-specific application. An architectural form is used as a model for defining HyTime elements, for instance a hyperlink. Any HyTime hyperdocument will then contain instances of "conventional" SGML elements and HyTime elements (conforming to a specific architectural form).

To properly evolve towards a HyTime repository therefore requires modeling of this architectural form construct; this leads us to extend the OOA repository architecture (cf. Figure 1) by:

- defining a new subsystem for the HyTime meta-DTD,
- enhancing the DTD subsystem model for identifying HyTime elements,
- enhancing the SGML Document Instance subsystem for identifying instances of HyTime elements.

But even when these enhancement has been performed, a HyTime document imported within the repository merely consists of an SGML document. Its hypermedia semantic aspects are not recognized because HyTime elements are identified but their semantics are not interpreted. The above-defined enhancements therefore only concern the syntactical layer model.

- The HyTime repository model: the semantic layer

HyTime document semantic aspects are generated in a HyTime processing phase by querying the syntactical layer and interpreting it according to HyTime specifications. Modeling a semantic layer then requires the addition of two new subsystems called "HyTime Processing Services" and "the HyTime Document Semantics". The HyTime Processing Services subsystem represents HyTime specifications contained in each HyTime module. These specifications must be modeled to provide all the classes and methods required to perform

HyTime processing. The HyTime Document Semantics subsystem which represents document semantic aspects generated by the HyTime processing phase. This subsystem enables the hypermedia semantic constructs associated with each HyTime architectural form to be modeled.

Finally, the HyTime processing phase enables generation of a hyperdocument which is composed of a set of document instances (syntactical layer) plus a set of document semantic contents (semantic layer); each document instance being associated with its related semantic content. These document semantic contents, generated by HyTime processing servers, consist of HyTime constructs organized into a complex model (HyTime Document Semantics subsystem).

Figure 5 gives an overview of the HyTime repository model organization (except the DTD and HyTime meta-DTD subsystems). Two kinds of relationships between the semantic and syntactical layers may be identified in this figure: “has for semantic” between a syntactical HyTime element and its related semantic construct, and “has for structural content” between a hyperlink node and its anchor point within a document instance.

As in [14], this modeling strategy induces redundant storage for objects which have both syntactical and semantic aspects.

4 AN SGML/HYTIME DOCUMENT MANAGEMENT SYSTEM PROTOTYPE

Figure 6 gives the architecture of the Database Management System we are prototyping. Here, we show how the applications are connected to the universal SGML/HyTime Object layer we suggested. This layer is based on the O2 ODBMS [28].

The SGML schema is populated at parsing time. When an SGML document is loaded, an SGML parser returns to the ODBMS a sequence of information corresponding to the structure and content of the document. Numerous objects are instantiated for the declaration, the DTD and the instance. External entities sharing is managed. However, special attention must be paid to cases in which SGML entities contain a partial tagging completed by the document referencing it. Each SGML useful reference is converted into an OID (Object IDentifier). Thus, the ODBMS manages the necessary functionalities like object sharing, object locking or object deep copies in case of inconsistent modifications of shared entities. An SGML document manager, based on a catalog manager, manages system and public identifiers. Then, the ID/IDREF links can be translated into OIDs by running a specific method.

The HyTime processing is run in a second pass. It is in charge of resolving the locators and the links. As the ODBMS used do not offer object mutation, we chose a multiple inheritance strategy to make an SGML object also contain the HyTime semantic. The HyTime processing methods populate the BOS (Bounded Object Set) according to a BOS level (limited deepness of inter-connected documents) associated with the documents.

Quick developments were possible because of the object modularity and the integrated tools offered with the ODBMS. In the first application, we developed a navigation interface, based on displaying the database objects, starting from the persistent document set object. Moreover, a graph of the traversed object is interactively built and displayed.

The second application is based on the ODMG query language named OQL. It enables filters to be applied on the database. Unfortunately, this access is reserved for specialists who know the SGML/HyTime document structure and the SGML/HyTime object database schema.

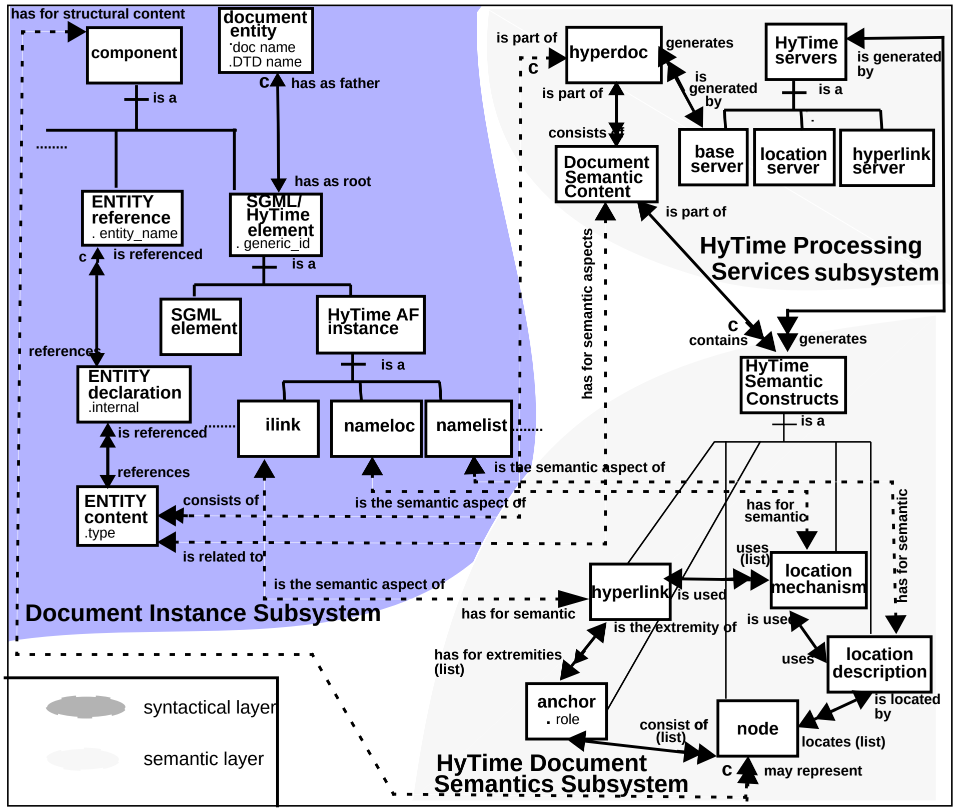

Figure 5. Evolution towards an HyTime repository model

Lastly, an HTML interface giving access to the SGML/HyTime documents, assisted by a fulltext search language, is currently under development.

5 CONCLUSION

At this stage, we have made an object-oriented specification of an object-SGML repository. We have also extended the specification with the HyTime domain for hyperlink management.

Concerning the prototype, we already developed:

- a generic object schema of the SGML standard,
- an SGML document loader,
- some graphical tools to navigate through and visualize SGML documents stored in an O2 database.
- an object schema of HyTime (hyperlink and location address modules)
- the first methods to compute HyTime Processing.

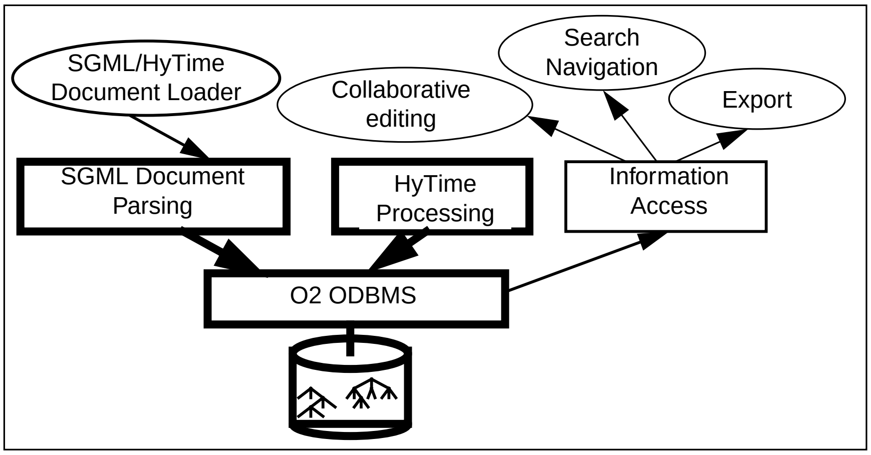

Figure 6. The HyO2 Document Management System architecture

Currently, we are extending our applications for the end-users interface. We offer hypertext-like navigation access, but we must work on an interface which hides the SGML/HyTime syntax and offers a query language based on SDQL [27], the language partially derived from DSSSLQuery and HyQ. This new language will run on top of OQL, the database query language. Some research work has already be done in the field of defining an SGML/HyTime query language on top of OQL [29,13].

We have obtained a very good foundation for an SGML/HyTime repository. Concerning HyTime, we are convinced that the HyTime concepts fulfil our hypermedia needs, but we are still waiting for the HyTime tools to come to the market.

Last but not least, we must improve performance because we will test our prototype on a large amount of documents (many tens of thousands for EDF) and on large size documents (many tens of megabytes for Aerospatiale).

In the future, we will use the HyTime concepts for document versioning. We will study how to map a "HyTime versioning DTD" to the ODBMSs versioning management module. The variant and versioning implementation are to be done.

REFERENCES

1. ISO 8879: Information processing—Text and office systems—Standard Generalized Markup Language (SGML), October 1986. International Organization for Standardization.
2. P. François and P. Bazex. SGML/HyTime repositories: Requirements and data modeling using object-oriented database concepts. Database and Expert Systems Applications (DEXA 95), 1995.
3. P. Futtersack and C. Espert. Electronic library system at Electricité de France: A case study using object technology. SGML Europe, 1995.
4. S.J. Mellor S. Shlaer, *Object Life-Cycles: Modeling the World in States*, Yourd Press, 1992.
5. K. Aberer, K. Böhm, and C. Hüser, 'Extending the scope of document handling: The design of an OODBMS application framework for SGML document storage', Technical report, GMD-IPSI, (1993).

6. K. Aberer, K. Böhm, and C. Hüser, 'The prospects of publishing using advanced database concepts', *Electronic Publishing—Origination, Dissemination and Design*, **6**(4), 469–480, (1994).
7. A. J. Brown. Approaches to document management: Relational versus object-oriented databases. SGML Europe, 1995.
8. C. Hamon. Conception orientée objet d'une base de données éditoriale SGML—Implantation sur le SGBDOO O2. Thèse (INRIA), 1992.
9. P. François, 'Generalized SGML repositories: Requirements and modeling', *Computer Standards and Interfaces*, **18**, (1996).
10. J. André, D. Decouchant, V. Quint, and H. Richy. 'Vers un atelier éditorial pour les documents structurés. Congrès AFCET bureautique, 1993.
11. F. Chahuneau and M. Rodriguez. Innovative aspects of SGML processing in the ground based electronic library system. SGML Europe, 1993.
12. K. Aberer and K. Böhm, 'Storing HyTime documents in an object-oriented database', in *Third International Conference on Information and Knowledge Management*, pp. 26–33, New York, (1994). ACM Press.
13. G. Gardarin and S. Yoon, 'Modeling and querying structured hypermedia documents', in *Database Systems for Advanced Applications (DASFAA 1995)*, pp. 441–448, (1995).
14. J.F. Koegel, L.W. Rutedge, J.L. Rutedge, and C. Keskin. HyOctane: a HyTime engine for an MMIS. ACM Multimedia, 1993.
15. J. Conklin, 'Hypertext: An introduction and survey', *IEEE Computer*, 17–41, (September 1987).
16. E. Barret, 'Hypertext in context', in *The society of text: Hypertext, Hypermedia and the social construction of information*, Collection Information Systems, Cambridge, Massachusetts, (1989). The MIT Press.
17. K. Gronbaek and R. H. Trigg. Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. Seventh ACM Conference on Hypertext, Washington, 1996.
18. L. Rostek, W. Mohr, and D.H. Fischer, 'Weaving a web: The structure and creation of an object network representing an electronic reference work', *Electronic Publishing—Origination, Dissemination and Design*, **6**(4), 195–505, (1994).
19. ISO 10744: Information technology—Hypermedia/time-bases structuring language (HyTime), 1992. International Organization for Standardization.
20. J. F. Buford, 'Evaluating HyTime: An examination and implementation experience', in *Seventh ACM Conference on Hypertext, Washington*, (1996).
21. F. Duluc. Structure d'accueil de documents hypermedia. DEA report (Aerospatiale—Institut National des Sciences Appliquées), 1995.
22. FMV (Swedish Defense Material Administration) Grund DTD. Sweden Cals Office, 1995. Version 1.10.
23. M. Biezunski. Modeling hyperdocuments using the topic map architecture. Second International Conference on Applications of HyTime, August 1995.
24. J. L. Sanson. EDF electronic library project: Data modeling with HyTime/CApH. Second International Conference on Applications of HyTime, August 1995.
25. S. J. DeRose and D. G.Durand, *Making Hypermedia Work*, Kluwer Academic Publisher, Boston, 1994.
26. C. F. Goldfarb, *The SGML Handbook*, Clarendon Press, Oxford, 1990.
27. ISO 10179: Information processing—Text and office systems, Document Style Semantics Specification Language (DSSSL), 1996. International Organization for Standardization.
28. O2. O2Technology Inc, 1995. Reference manuals.
29. V. Christophides, S.Abitetoul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. 13th ACM SIGMOD Conference, 1994.