Invited Paper

Multimedia Document Databases: Representation, Query Processing and Navigation

Ravi S. Kalakota* and Andrew B. Whinston*

Abstract

Information systems for application areas like office automation, customer service or computer aided manufacturing are usually highly interactive and deal with complex document structures composed of multiple media formats. For the realization of these systems, nonstandard database systems, which we call document databases, need to handle different types of coarse-and fine-grained document objects(like full-text documents, graphics and images), hierarchical and non-hierarchical relationships between objects(like composition-links and cross-references using hypertext structures) and document attributes of different types such as formatting/presentation information and access control. In this paper, we present the underlying data model for document databases based on descriptive markup languages that provide mechanisms for specifying the logical structure(or schema) of individual documents stored in the database. We then describe extensions to the data model for supporting notion of composite structures("join" operators for documents) --composition and hyperlinking mechanisms for representing compound documents and inter-linked documents as unique entites separate from their components. Furthermore, due to the interactive nature of the application domains, the database system in conjunction with clients(or browsers) has to support visual navigation and graphical query mechanisms. We describe the functionality of a new user interface paradigm called HyBrow for meeting the above mentioned requirements. The underlying implementation strategy is also discussed.

* Department of Management Science and Information Systems, University of Texas at Austin, Austin, Texas 78712

1. Introduction

The importance of data as a valuable and persistent information asset of organizational knowledge and functioning is widely accepted as the basic axiom when designing information systems. Databases have been developed to store, manage, and allow access to essential data for business processes. Modern DBMSs are being used to manage a wider variety of data types and to provide integrated support for various business processes and functions [Stonebraker & Kemnitz 1991]. However, standard RDBMS technology does not adequately meet the needs of certain non-traditional application areas like electronic publishing, office automation, customer service or computer aided manufacturing. These application areas typically use documents as the basic data type for organizational work related activities. For instance, valuable information, often relevant to organizational problem-solving and decision-making, is not stored in corporate databases, but instead contained in on-line documents such as memos, electronic mail, manuals for standard operating procedures, technical manuals, regulatory and legal documents. To obtain maximum benefit from the voluminous amounts of on-line documents, data management techniques need to be adopted that promote coherent document structuring, distribution and manipulation.

In many organizations, large amounts of on-line documents cannot be easily accessed or manipulated by end-users because they are stored on different hardware platforms under diverse operating systems, application software or use format specifications that are not interchangealbe or interoperable. The *document interoperability* problem is compounded further by the fact that document content is not simple text but may contain a multitude of data types ranging from formatted text(e.g. Postscript, PDF), graphics (e.g. CGM, GKS), images(e.g. bitmap, GIF or JPEG), video(e.g. QuickTime, MPEG) and audio(e.g. MIDI(hi-fi stereo), WAV) segments. In addition to document interoperability, organizations planning on using on-line documents are faced with two problems : the need to *reformat* documents for different media--print, display, etc., and perhaps even more important, the need to access and retrieve information "trapped" in an ever-growing repository of electronic documents. To solve these problems, what is required is a way to create a schema structure for documents that provides a means for identifying or "tagging" the information elements of documents with great flexibility so that the structure of the information in the document and the relationships of its parts may be described with the clarity required to allow automated assembly, disassembly, modification and reassembly of the information elements.

Many computer users engage in the creation of what can be called compound documents-documents with parts containing various media, such as text, tables, movies, sound and graphics in a variety of file formats[OpenDoc 1993). Every part contains data—for example, text parts contain characters, graphics parts contain lines and shapes, spreadsheet parts contain spreadsheet cells with formulas, and video parts contain digitized video. The particular type of data that each part contains is known as the part's intrinsic content. In addition to its intrinsic content, a part may contain other parts. Currently each medium requires users to work in different ways, and often in separate applications or editors, demanding a labor-intensive series of actions to view and manipulate the information in the document part. This lengthy and cumbersome process tends to be error-prone and frustrating as there are no formal rules for aggregating different document parts to create consistent structures. For example, every document has a single part at its top level, the root part, into which all other parts are embedded in a hierarchical structure. Clearly, we need techniques for specifying the intrinsic content of document parts and the logical structure of document with respect to the various parts.

In section 3, we deal with the task of specifying a document schema structure for specifying the intrinsic content(data model) of any document. We use Standard Generalized Markup Language (SGML)[Bryan 1988, Goldfarb 1990, ISO-8879 1986] for specifying the within document structure. SGML comprises a single set or rules(called descriptive markup or tags) that specifies the structure of a document, independent of its format. Each class of SGML documents has an associated Document Type Definition (DTD) file that contains lists of allowed tagged elements and their hierarchical relationships. The DTD file explains the SGML tagging structure and can be used by a parser to check and flag any noncompliant tags. This programming-like aspect of SGML gives it the potential to operate on a more profound level than any other document standard developed for document content specification.

Document interoperability is further made difficult by the fact that the notion of a document as an entity with clear boundaries is fast becoming antiquated. With embedded hypertext links, a document can no longer be considered an isolated entity but a network of interlinked textual and multimedia information[Halasz & Schwartz 1994]. The ability to browse around a network by following the links from node to node is a defining feature of hypermedia document [Conklin 1987]. Hypertext is important as the page metaphor which is the basis for thinking about traditional documents(e.g., publishing or word processing) could prove to be a hindrance in working with electronic documents. Scrolling up and down pages is certainly out of the question for users who plan to spend lots of time referencing very long documents. The page metaphor is not appropriate for navigating through hypertext document structures, where the user traverses links between document elements which could be as fine grained as words or paragraphs or coarse grained as document parts mentioned earlier.

The basic hypermedia document network has only two primitive constructs : document nodes and links. Although this model has been widely adopted for building simple hypermedia structures[Goodman 1986] and more recently distributed hypermedia [Berners-Lee et al. 1992], it is insufficient for building complex information systems capable of serving the organizational needs. In particular, the model lacks a composition mechanism for building composite structures, i.e., a way of representing and dealing with group or collections of document nodes and links as unique entities separate from their components[Halasz 1988]. The goal of composite structures is to reduce large document spaces into more manageable sets using abstraction mechanisms such as aggregation, association or classification hierarchies. Also, navigation through a large document network can be made simpler with the notion of composite structures. For instance, KMS[Akscyn et al. 1989] uses a top-down, stagewise refinement approach to organizing material in the hypertext database called "hierarchical skeleton". The resulting hierarchical "skeleton" in the database helps users build a coherent mental model of the database. They can remain oriented when navigating because they can always see whether they are selecting a hierarchical link or a cross-reference.

In section 4, we deal with hypermedia extensions of the data model specified in section 3. Our emphasis is on specifying composite structures that permit abstract(or higher level) view of the document space than the traditional node-link model of hypertext. We investigate the notion of a document classification hierarchy(by hierarchy, we mean a partial order) that describes collections of documents and the containment relationship among these collections. In particular, we examine how the operations of aggregation and association can be used to organize and structure the multimedia document space. Aggregation[Botafogo & Shneiderman 1991] is a way of grouping related documents or document parts that may be interlinked. Aggregation implies a much looser relationship than a type(part/whole) hierarchy. It implies an organization structure in which the participating entities allude to each other but remain essentially independent. Association deals with creating names which are persistent. For example, documents which are distributed around the organization but related to a customer must all be collected under one association called "CUSTOMER". Our goal is to develop an understanding of the operators needed for building composite structures that reduce the complexity of document space and enable more effective user interfaces for querying and browsing of unfamiliar collection of documents.

Often users can describe exactly what information they are looking for, but simply cannot find it in the network. Excessive reliance on navigation for document access can be time consuming and problematic in large, unfamiliar networks. Navigational access using node-to-node links is probelmatic because users tend to get lost while wandering around in the network looking for some target information. To complement navigation, effective access to information stored in a hypermedia document network requires *query-based access*. Two types of query mechanisms are possible in document networks : content-based and structure-based queries[Halasz 1988]. In content-based query, all document nodes and links in the network are considered as independent entities and are examined individually using regular expression or string matching functions for a match to the given query[Salton 1991]. Structure-based query examines the individual document structure or network structure for sub-networks that match a given pattern[Arnon 1992]. An incremental solution to the navigational problems would be to improve and augment the existing navigation tools by interleaving browsing and querying. In other words, a more fundamental solution is to augment navigation by a query-based access mechanism. With such a mechanism, the user could formulate a query encapsulating a description in the network. The structure that is returned is then browsed by the user to further narrow the possibilities.

In section 5, we present content and structure based queries on composite structures and within document content. As pointed out earlier, query facilities which combine aspects of both content search and structure search will be needed for browsing through document networks. For these reasons, we have taken a much more general approach to repository access, called Query by Browsing, which we describe in section 5. In Figure 1, we show how the three various issues in document database management that we have presented fit together.

The rest of the paper is organized as follows. We present relevant background work and related research in Sesction 2. The data model for supporting the complex application requirements is presented in Section 3. In Section 4, the hypertext extensions to this model are discussed. We then present the new user interface paradigm called HyBrow where browsing and querying are inter-

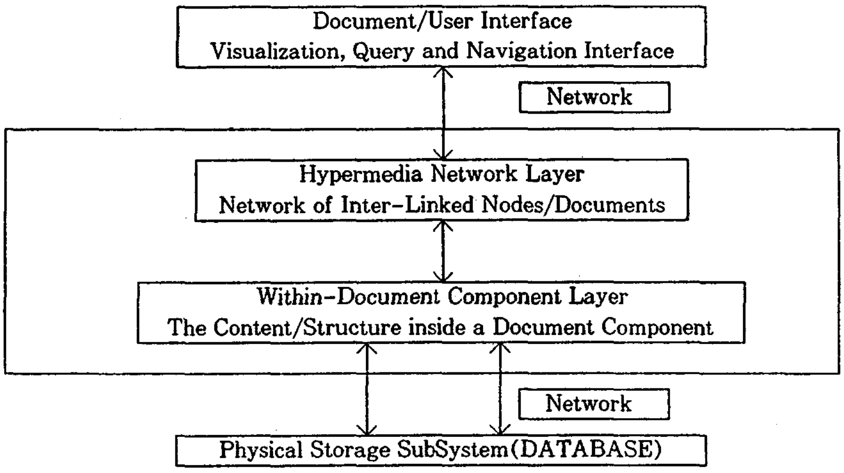

Document/User Interface
Visualization, Query and Navigation Interface
Network
Hypermedia Network Layer
Network of Inter-Linked Nodes/Documents
Within-Document Component Layer
The Content/Structure inside a Document Component
Network
Physical Storage SubSystem(DATABASE)

Figure 1 : Structure of Document Management System

leaved to create an effect known as the Query by Browsing model. In this section, we describe the mechanisms by which query parsing(initial preprocessing), query optimization and query execution are handled to accomplish the HyBrow paradigm. Finally, in the conclusion, we briefly describe research extensions and implementation challenges.

2. BACKGROUND

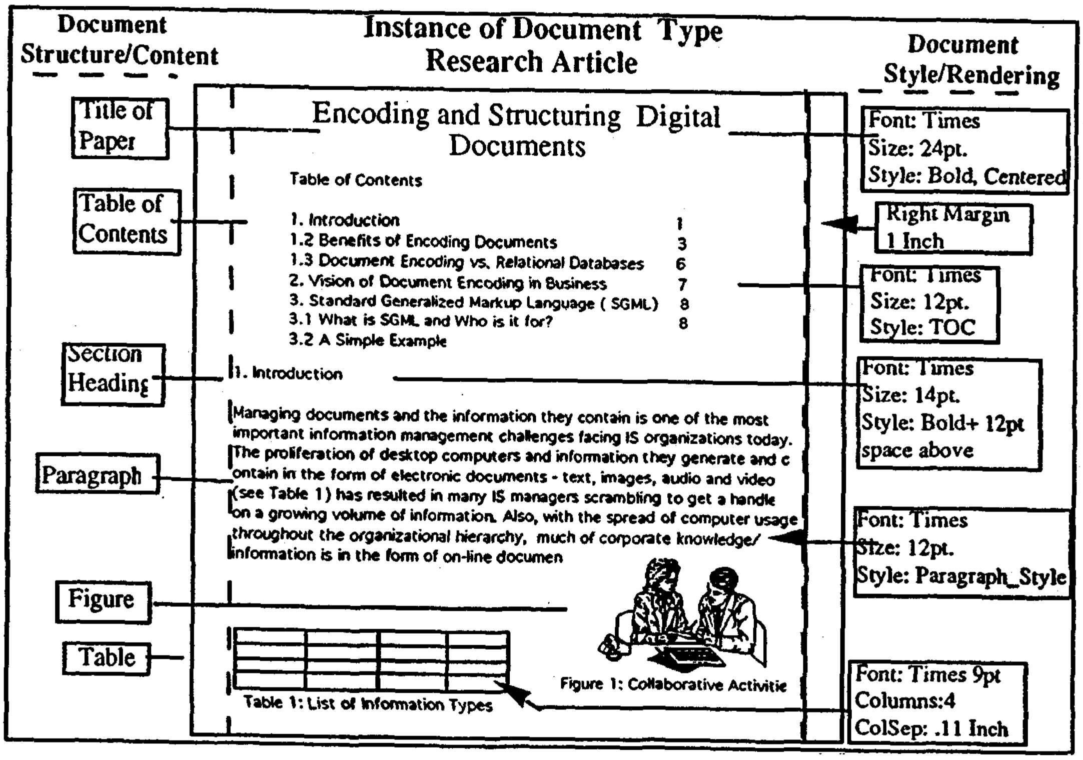
2.1 Descriptive[1] Markup

Conventional documents are often structured and stored in such a way that document content is hard to access and manipulate. One method for solving this problem is to separate the logical structure of the document from the physical document structure (or formatting)(see Figure 2). The class of documents where this separation takes place is known as structured documents. Structured documents differ from traditional documents(e.g. word processing files or document imaging) as they contain descriptive markup that describes the logical structure of the document. This logical structure can be effectively utilized for describing the topology of multimedia documents containing several components-formatted text(e.g. Postscript), images(e.g. JPEG), video(e.g. MPEG) and audio segments.

1) Descriptive markup is also known as Generalized markup. Markup is so named because of its resemblance to the markings that copy editors make on drafts of paper documents.

Traditionally, the area in which structured documents have had the most relevance has been in the formatting, display and publishing of electronic documents. Three important development here are the Open Document Architecture(ODA)[ISO-8893 1986], Digital Equipment Corporation's Compound Document Architecture(CDA) and Standard Generalized Markup Language (SGML) standards. ODA and CDA are largely concerned with the interchange of office documents across different platforms, whereas SGML provides a framework for developing document data models. Both ODA and CDA provide a set of standards for the interchange of complex documents made up of text, images, and graphics among computer platforms and applications. Compound document architectures encode documents using in-memory arrays called aggregates. Aggregates can be used to represent audio, graphics, text, and video, as well as a document's physical formatting, logical organization, and text styling. Aggregates can be processed as a document or as an information database, such as a parts list, an index, a glossary, or a multi-media document. SGML accomplishes document encoding using a different techinque than CDA or ODA. Rather than dynamically creating new and different data types, as do CDA and ODA, SGML uses special character sequences known as markup tags to embed control information within the text stream. Markup tags can separate a document's logical elements or specify processing functions to be performed on them. In essence, these standards, particularly SGML, enable us to specify the logical structure of documents (specified using descriptive markup), separately from the formatting information(specified using procedural markup). Procedural markup languages(e.g. TeX, LaTeX, TROFF, SCRIBE) provide a set of methods to express how a document should be processed by the printers or the document processing systems. For instance, in TeX [Knuth 1989] backslash means that subsequent input is TeX instructions. These markup languages offer additional constructs for building more abstract macros. For instance, LaTeX[Lamport 1986] through macros allows itemized lists, instead of indents, item numbering, among other things.

Structure or descriptive markup is based on two postulates : (i) it should describe document structure and other attributes rather than specify processing to be performed on it as structure markup need only be done once for every document and will suffice for all future processing[Goldfarb 1990]; (ii) Markup should be rigorous so that techniques used for processing software, e.g. parsing and database integrity validation can be used for processing documents as well[Goldfarb 1990]. Markup is entered by users in terms of codes or other instructions via electronic typesetting programs, which in simple cases is the editor. An example of markup code is TROFF's[2] or WordStar's ".ce" for "center the following line". WYSIWYG[3] editors hide markup codes from us by showing text as it appears on output. Four types of markup are : punctuation(spaces, punctuation), presentational (layout, font choice), procedural(formatting commands), and descriptive(mnemonic labels for document elements)[Coombs 1987]. Markup is done using markup languages and markup-aware text editiors. Unlike artificial languages, markup languages have to deal with embedded data, and contain rules for what is markup and what is content. Figure 2 illustrates differences between document structure and markup that represents formatting information.



Encoding and Structuring Digital Documents

Table of Contents

1. Introduction 1
1.2 Benefits of Encoding Documents 3
1.3 Document Encoding vs. Relational Databases 6
2. Vision of Document Encoding in Business 7
3. Standard Generalized Markup Language (SGML) 8
3.1 What is SGML and Who is it for? 8
3.2 A Simple Example

1. Introduction

Managing documents and the information they contain is one of the most important information management challenges facing IS organizations today. The proliferation of desktop computers and information they generate and c ontain in the form of electronic documents - text, images, audio and video (see Table 1) has resulted in many IS managers scrambling to get a handle on a growing volume of information. Also, with the spread of computer usage throughout the organizational hierarchy, much of corporate knowledge/ information is in the form of on-line documen

Figure 2 : Separating Document Structure from Content and formatting

2) Trogg is a Unix based text formatter program [Kerningham 1974] which has widely been used in the Unix community, especially before graphical user interfaces when editing was first done with tools such as vi, and the formatting for printout had to be done as a separate process(batch). Nroff is also a venerable text-processing program, usualy supplied with Unix-based computer systems.

3) What You See Is What You Get.

Descriptive markup is not tied to formatting or printing capabilities. It focuses on describing underlying structure of documents and says nothing about formatting. With generalized markup, users tell the system what the document has, rather than how it should look, and this is often done by putting a label(i.e. tag) around the text. There is a clear correlation between tags and what things look like. However, the tags do not determine how the actual output will look-whether the "Heading" is to be printed or displayed as 14-point or 18-point. This is a matter of style, and style is the job of the output specification a separate file that can be attached to the document. This "freeing content from formatting information" makes it possible to use the same document for different delivery media : document instances can be processed for Braille delivery, on-line screen delivery, audio delivery, machine-to-machine delivery, paper delivery, or all of these purposes.

Descriptive markup is being primarily done using SGML[ISO-8893 1986; Goldfarb 1990]. SGML is entirely descriptive, leaving procedures and formatting to the devices that read and interpret the tagging systems defined in a document-type definition. Section 2 describes this in greater detail.

2.2 Hypertext documents vs. Conventional Documents

To understand why hypertext documents are attracting attention, one must understand how a hypertext "document" differs from a conventional paper document. In most conventional paper documents physical structure and logical structure are closely related. Physically, the document is a long linear sequence of words that has been divided into lines and pages for convenience. Logically, the document is also linear : words are combined to form sentences, sentences to form paragraphs, paragraphs to form sections, etc. If the document has a hierarchical logical structure, as do many expository documents such as books, that hierarchy is presented linearly : the abstract or overview comes first, followed by the introduction, the first chapter, the second chapter, etc., untill the conclusion. This linearity is easy to see if one imagines the hierarchical structure represented as an outline, with the sections of the document appearing in the same order as they normally do in the outline. Such documents strongly encourage readers to read them linearly, from beginning to end follwing the same sequence.

A few conventional paper documents-encyclopedias, dictionaries, and other reference works-separate logical structure from physical structure. Physically, these documents are linear sequence of independent units, such as articles on specific topics or entries for individual words. Logically, they are more complex. The reader seldom reads such documents from beginning to end, but rather searches them to locate the article or entry of interest(a form of random access), and then reads that portion sequentially. However, the reader is likely to encounter various cross reference to other entries while reading as well as a list of "see also's" at the end of an article. To follow those pointers, the reader must locate the appropriate volume, find the appropriated entry, and then the relevant portion.

The logical structure of reference and other similar documents is, thus, more complex. They have a sequential structure that aids search, but the logical path of the reader is a network that can criss-cross the entire document or set of documents, from one item to another, to another, etc. Such documents are more flexible but they are also cumbersome, particularly when they appear in large, multi-volume formats.

Hypertext electronic documents provide most of the flexibility of reference works as well as add a number of new features. Earlier, we described a hypertext as a document in which information is stored in nodes connected by links. Each node can be thought of as analogous to a short section of an encyclopedia article or perhaps a graphic image with a caption. The links join these sections to one another to form the article as a whole and the articles to form the encyclopedia. These links are usually shown for each node as a "start" link pointing to the node just read and a set of "end" links that indicate the(usual) multiple node which one may select to read next. Many systems also include pointers embedded in the text itself that link a specific portion to some other node or portion of text. Thus, one moves from node to node by selecting the desired "to" link, an embedded cross-reference link, or the "from" link to return to the previous node. For many documents, the "to" links can be thought of as organizational. Collectively, they frequently form a hierarchical structure analogous to the hierarchical logical structure links cross the main organizational structure. While we can establish a rough analogy between the two, hypertext documents are much more flexible than conventional documents. for example, one can read the hypertext article just as one reads the conventional paper article by first reading the overview node, then the first section node(s), the second section, etc. However, one can also read the sections in different orders. Hypertext documents are also much more convenient. To follow the cross-references in a modern encyclopedia often means moving among many volumes. Readers do it, but it is a slow, frequently laborious, task.

While hypertext provides greater flexibility than conventional documents, its power and appeal increase dramatically when it is implemented in computing environments that include networked microcomputers and workstations, high-resolution displays, and large on-line storage. While hypertext systems can deliver the next node in less than a second and form a much larger body of information that might take thousands of volumes in print. While conventional publications are limited to text and graphics, hypertext nodes offer sound, video sequences, animation, even computer programs that begin running when the node in which they are stored are selected. While the organizational and cross-reference structures of conventional documents are fixed at the time of printing, hypertext links and nodes can be changed dynamically. Information in individual nodes can be updated, new nodes can be linked into the overall hypertext structures, and new links added to show new relationships. In some systems, users can add their own links to form new organizational structures, creating new documents from old.

2.3 Document Search and Query

The serarch and query process for documents can be divided into two broad classes : content search and structure search [Halasz 1989]. We can partition these two classes to be more precise about the type of document space being searched in, namely with-in document (intra-document search) or the network of documents(inter-document search)(See Figure 3). Content search for inter-document links(i.e., all nodes and links are treated independently and examined for a match to the given query) has been addressed to a very limited extent (mostly for single machine systems) by hypertext research. Intra-document content search has points in common with information retrieval(IR) techniques that have been developed over the last 30 years to access bibliographic databases and electronic card catalogs for libraries.

	CONTENT	STRUCTURE
INTRA-DOCUMENT	Key Word Indexes Vector Space Probabilistic Models	Descriptive Markup based Tree Transformations
INTER-DOCUMENT	Indexing Aggregate document collections	Lattice-Based Associative Structures

Figure 3 : Search and Query Space

In traditional IR applications, each document is indexed by an expert, who specifies a set of index terms from a controlled vocabulary(such as the Library of Congress Subject Headings or a thesaurus of index terms) which describe the subject material of the document. In on-line IR system these manually assigned index terms are supplemented with keywords automatically extracted from the titles, abstracts, and(when available) the full text of the document. Commonly, the individual words of controlled index terms(which may be descriptive phrases or classification numbers) are also treated as keywords. A user who wishes to find documents in a collection that might be relevant to his interests, specifies his request by indicating a Boolean combination of search criteria on specific fields, e.g. :

Title Keyword="Wall Street" and (subject="derivatives" or subject="brokerage")

System with this sort of Boolean Query language include commercial bibliographic search services, such as LEXIS/NEXIs and DIALOG, and a majority of on-line library catalogs. More advanced retrieval methods have been developed that replace Boolean logic with more sophistricated matching techniqués. These techniques, such as the vector space and probabilistic models of IR, attempt to rank the documents in the database in order of their similarity, or probability of relevance, to a given natural language query[Slaton 1991]. Systems using these methods include 13R[Croft 1987] and WAIS[Kahle 1991].

There are Three problems with applying the traditional IR paradigm to documents :

1) Traditional IR depends on the existence of a human(or a program) to index the document by specifying a collection of keywords. These documents have traditionally been text and it is not clear how to specify such keywords about images, video or mathematical documents(e.g., spreadsheets). The problem of searching through heterogeneous "mixed"-different formats and data types-document collections remains a research challenge.

2) As stated earlier, not all document objects will be text. Many other kinds of documents, such as compound documents, must be stored and indexed. It is not clear how to apply the traditional IR paradigm to a hierarchical document structure where several documents may be embedded in the root document. For instance, a business report may consist of graphics, spreadsheet and text objects. These objects may be nested, i.e., in turn contain other objects. Structural navigation of compound documents has not been addressed.

3) Traditional IR does not address the

problem of hypertext retrieval, i.e., network of interlinked documents. In hypertext, the user can navigate over the information by following the links(or cross reference) in the document. This form of associative navigation is context dependent and if not carefully organized can produce complex, disorganized tangles of haphazardly connected documents.

Structure search is useful when browsing thorugh a document or navigating through hypertext networks. However, there has been relatively little work in the area of structured document retrieval. Structure queries allow the user to specify a preferred region which is to be explored further. The result of a structure query will yield a sub-network(in the case of inter-document networks) or a sub-tree from the hierarchical SGML document structure.. The MULTOS project is based on ODA[ODA 8893 1986] and as described in [Bertino-Lee et al. 1988] only provides rudimentary structure based search. The OED project provides more general retrieval capability than conventional systems, but is restricted by the database search command language "PAT" used to express the queries[MacLeod 1991]. In this paper, we address the issue of navigation and query of document structure in section 5. But, in order to do structure-based search, we must first understand the document data model that provides a logical schema for documents.

3. DOCUMENT DATA MODEL

An important issue in designing document management systems concerns the design of a suitable data model for structured documents. The data model provides a formal structure for the storage, retrieval and construction(or restructuring) of documents. To better understand document data models, we need to differentiate between various structures that constitute the data model. The most common distinction is between internal representation(or syntax) structure and the outward appearance or visual geometry. Other researchers have characterized this distinction as : abstract objects versus concrete objects[Kimura 1984], document model versus output model[Furuta 1987], logical structure versus physical structure [Peels 1981] and logical structure versus visible appearance[Lamport 1987].

Structured document models can also be described in terms of document schemas, instances and layouts. These are related in the following manner :

Document Instance=Schema+Layout+Content

A schema describes the possible structure and data values corresponding to a class of documents. The patterns of structuring follow the dictates of some domain(e.g., Finance, Accounting) to which the document belongs. Two types of schema strcutures are used in describing a document : Abstract Schema and Concrete Schema. The Abstract Schema describes the logical structure of the document. For example, an Abstract Schema for a text-book describes a logical structure of index, chapters, sections, references and appendices. Chapters may be structured as title, chaper_introduction, chaper_body and conclusions. Chapter_Body is further structured in sub-sections, headings, paragraphs, graphics and so on. This is one Abstract Schema for a book. There could possibly be parallel Abstract Schemas for a class of documents known as text-book. For example, we can have an Abstract Schema that defines text-book for the blind, text-book for the K-12, text-book for colleges and so on. The Abstract Schema does not provide the details about the content structure. This is the task of the Concrete Schema that defines phrase structures and well-formed sentences for that particular task domain. The Concrete Schema also defines the character set(e.g., ASCII, ISO 10646 etc.), acceptable notations(e.g., Math, Image, Video formats etc.) and syntax rules(e.g., Identifier names must be less than 12 characters in length). For every logical structure defined by an Abstract Schema there could be multiple definitions of Concrete Schemas.

Each schema has one or more layout associations. The outward appearance of the document depends on the formatting instructions provided and the medium on which the document is being rendered. A layout is a mapping(one-to-many) between the actual content(document instance) and the output device, such as printer, monitor, Braille or audio for the visually impaired. Visual formatting renders the document structure on a two dimensional dispaly(paper or monitor) using the specified formatting rules. The visual layout helps the reader recreate, internalize and browse the underlying document structure. The ability to selectively access portions of the display, combined with layout, enables multiple views. For example, a reader can first skim a document to obtain a high-level view and then read portions of it in detail. We assume that for the purpose of describing the layout mapping a full-fledged formatting system[Adler 1994] is available.

With new media types, layout is not restricted to visual formatting and may include audio formatting, which renders information structure in a manner attuned to an auditory display. The influence of schemas on layout processing for media such as video and audio is a new area of research where little has been done. The traditional paradigm in text-based documents is that the reader is active, while the display is passive. This active-passive role is reversed by the temporal nature of audio and video, that is, information flows actively past a passive listener or viewer who has little control, except for actions such as STOP, PLAY, FAST FORWARD or REWIND. This prohibits multiple views, that is, it is impossible to first obtatin a high-level view and then "look" at details of such documents. However, how the layout mapping is to be specified for video and audio is beyond the scope of this paper and is described elsewhere[Raman 1994].

The formal model developed in this section using SGML deals with documents as data structures only ; that is, it describes documents in terms of schemas and instances. As described earlier, it is worthwhile to note that in SGML, the schema structure(provided by descriptive markup) is completely separated from the layout aspect(provided by procedural markup). The SGML document schema defines the logical structure and language/context of the document. It presents a collection of "grammar" rules for specifying a set of valid document instances. The syntactic definition for the SGML document schema can be formally defined using an extended Backus-Naur Form(BNF) as :

```
⟨SGML_Document_Schema⟩
    : : =⟨structure decl⟩ | ⟨process decl⟩
⟨structure decl⟩
    : : =⟨abstract_structure decl⟩ | ⟨concrete_structure decl⟩
⟨process decl⟩
    : : =⟨interface decl⟩ | ⟨layout_process decl⟩
⟨interface decl⟩
    : : =⟨view_process decl⟩ | ⟨query_process decl⟩
```

In this section, we will deal with ⟨structure⟩, namely the ⟨abstract_structure⟩ and the ⟨concrete_structure⟩. In the latter section, we will present how the ⟨process decl⟩ are derived from the ⟨structure decl⟩. For instance, the document schemas can be processed for answering queries(query processing);for effective visual presentation(layout processing);or for extracting and customizing a specfic view of the document(view processing).

3.1 Abstract Syntax

The abstract syntax, given as a set of production rules, defines the decomposition of the document into logical units or types. It is the core of the grammar and neither the concrete syntax nor the visual geometry can be difined without reference to it. In this paper, we present a model of abstract syntax which is a variation of the operator-phylum model as used in MENTOR[Donzeau-Gouge 1984], GANDALF[Notkin 1985] and the Synthesizer Generator[Reps & Teitelbaum 1989]. The operator-phylum model[Donzeau-Gouge 1984] was invented for use in structure-oriented editors. The user of such system works only with the logical structure and writes by progressively expanding the parse tree all the way down to the leaves. This careful demarcation of abstract syntax and concrete syntax is not found in BNF[Naur 1960] or phrase structure grammars[Chomsky 1956] and is useful for developing higher level structures that prove to be very handy for querying and browsing.

In the operator-phylum model, operators represent logical decomposition by listing a sequence of phyla(borrowing the term from abstract algebra). Phyla define logical types and are arranged in a hierarchy of subtypes and supertypes. At the base of the hierarchy are leaf phyla, such as INTEGER or STRING. A phylum whose element subset is empty is called a terminal phylum and one whose set of subphyla is non-empty is called a category. Although SGML has no subtypes, some of the effect of having phyla can be achieved by using groups in ELEMENT declarations, particularly if the group is defined via a parametric ENTITY, which gives it a label similar to a category. For example, these declarations create the illusion of supertype :

```
〈!ENTITY      %      Link
    "URL | URN | URC | Hytime-clink"〉
〈!ELEMENT MultiMediaObjectLink-
    O((StratObject, %Link;, EndObject) &
    LinkLabel)*
```

The first declaration implicitly defines a logical type called "Link" with subtypes of the various hyperlink standards. The second declaration uses this logical type by defining a supertype called "MultiMediaObjectLink" which utilizes the earlier type definition. Now let us examine the abstract syntax structure of an electronic catalog which is being designed for use in electronic commerce. The catalog structure is generic enough to describe a wide range of components, in this case a wide range of electronic components-resistors, capacitors, transistors etc.-and devices, both mechanical-fans, motors etc.-and electrical components-power supply etc. In order to simplify conceptual modeling, we can think of information about each component as being stored in one document. One can then describe the abstract syntax of this document as a tree-structure with each node containing a hierarchy of structural types. For instance, we can broadly distinguish several nodes(or logical units) in the electronic componont document depending on the information being modeled : administrative, engineering, manufacturing, technical support and purchasing. Each logical unit, in turn, hosts a hierarchy of sub-types that describe the logical unit in increasing detail. See Figure 4.

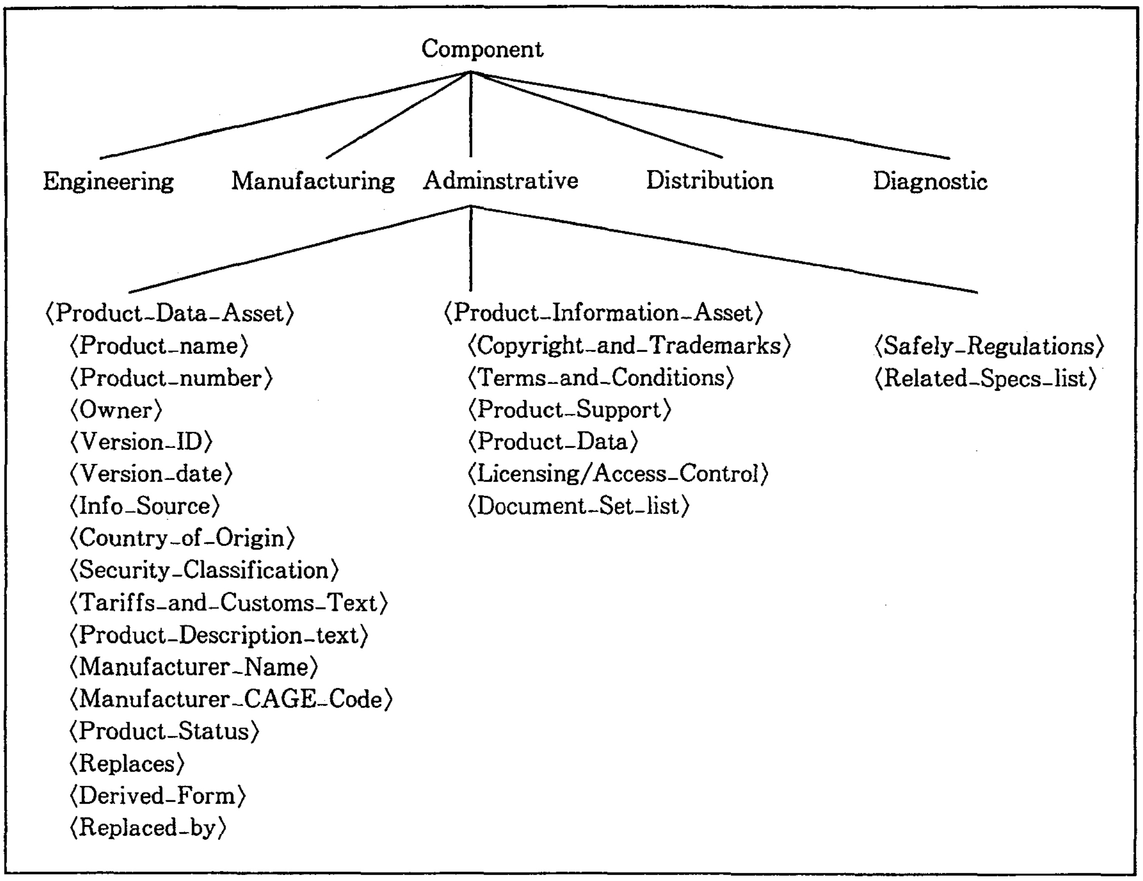


Figure 4 : Logical Structure of a Component Document

The logical structure presented above can be easily mapped on to SGML language statements as illistrated by the segment below :

```
⟨!ENTITY% DocumentType "Electronic Catalog"⟩
⟨!ENTITY% DocumentElement "Admin-Info,
    (Engineering-Info & Manufacturing-Info & Diag-
    nostic-Info & Distribution-Info?")⟩
⟨!ENTITY% Data-Assets "Ower-Id & Product-De-
    scription & Manufacturer-Name")
⟨!ELEMENT % DocumentType; OO(%
    DocumentElement;)
⟨!ELEMENT Admin-Info -O(Product-Data-Asset &
    Product-Information-Asset & Product-Safety-
    Asset))
⟨!ELEMENT Product-Data-Asset) -O(Product-
    Name, Product-Id, (%Data-Assets;)
    Product-Information-Asset & Product-Safety-
    Asset))
```

In order to customize information to address different business needs, for instance, providing technical data to customers, usage in advertisements etc., we must have an underlying repository of component information encapsulated in documents. To create these document instance, the phyla of the abstract syntax must be transformed into production rules with terminals or literals.

3.2 Concrete Syntax

As pointed out earlier, the abstract structure describes the logical properites and the concrete syntax provides implementation specific properties. Concrete structure specifications represent lower of the abstract syntax tree ultimately terminating in phrase structures. The concrete syntax defines the interplay between the abstract syntax and visual geometry. It provides the literals used to construct the logical structure and indicates the lexical ordering of the elements. Here we present details of the abstract structure presented earlier. It is important to realize that for any abstract structure there can be many *corresponding concrete structures*. Take for instance, Graphic Notations and Math Notations. This abstract entity and be implemented in a number of ways as indicated below.

```
〈!-Graphic. Notations defines the names of the graph-
   ic notations supported.-〉
〈!ENTITY % Graphic_Notations
   "Image | Vector | GIF | EPS | PICT | CGMChar |
   CGMCLear | BMP | MET | TIFF | CDR"〉
〈!-Math.Notations defines the names of the math and
   equation notations supported.-〉
〈!ENTITY      %      Math - Notations
   "TeXMath | eqn | ISONath | Mathematica"〉
```

To understand the concrete syntax, examine the fragment of a concrete syntax pertinent to an electronic catalog. The phyla "ProductData" is enhanced with a series of production rules with literals.

```
〈!ELEMENT ProductData···(ProductNumber, (Own-
  ers?
  | Versions? | InformationSource?
  | CustomsInformation? | ProductDescription?
  | ManufactureInfo? | ProductStatus?
  | ProductSupport? | ReleastInformation?
  | Desc?)*)〉
〈!ELEMENT ProductNumber-O(#PCDATA)*〉
〈!ELEMENT Owners-O(Enterprise | Person)+〉
〈!ELEMENT ProductSupport-O(SalesContact
  | Marketing Contact | TechnicalContact)*〉
〈!ELEMENT (SalesContact | MarketingContact
  | TechnicalContact) OO(Enterprise | Person)+〉
〈!ELEMENT EnterpriseName OO(#PCDATA)*〉
〈!ELEMENT Person-O(Name, Address?)〉
〈!ELEMENT Name OO(NameBlock? | (Last, First?,
  Middle?, JobTitle?, Desc?))〉
〈!ELEMENT  (Last | First | Middle | JobTitle | Desc)
  OO(#PCDATA)*〉
```

The literals in SGML range from # PCDATA-parsed character data, # CDATA-character data where nested SGML *markup is not permitted, to* # RCDATA-replaceable character data containing text, character references that resolve to character data. SGML provides four techniques for minimizing the number and length of document markup tags : tag omission, tag shortening, tag grouping and automatic tag recognition[Bryan 1988]. Of these the most used is tag omission. This is indicated by the "O". If the strat tag can be omitted, a hyphen "-" is placed in the first character. The following symbols are used extensively in the SGML grammar and an explanation is provided so that the reader can understand the usage. With this understanding the SGML fragment presented above is fairly straightforward.

, - All must occur in the order shown & - All may occur in any order | - One and only one must occure	? - Optional(0 or 1 time) + - Required and repeatalbe (1 or more times) * - Optional and repeatable (0 or 1 times)
Connectors	Occurence indicator

4. COMPOSITE STRUCTURES

In the previous section, we have outlined the SGML-based data model for describing the intrinsic document content(or within document structure). Although the SGML data model does capture the internal structure of documents, it still comes up short when trying to represent relationships between documents and semantic generalizations(such as classification, aggregation or association) of document behavior. In this section, we present an extension to the data model for supporting more shphisticated inter-document relationships using the notion of composite structures. A composite structure uses the composition mechanism to deal with a set of nodes or links as a single object. The focus in this section is on modeling composite structures comprised of document nodes as seen by the application and the user.

Inter-document relationships have promarily been accomplished using the simple node-link model provided by hypertext. Nodes are the basic information containers. Links are used to connect two nodes which have associated text or ancillary information. Links, therefore, are the mode of navigation in a hypertext network. Nodes in hypertext can come in two varieties : typed and untyped. An untyped node is a box for information. It has no label or descriptor, and therefore may contain anything. Examples of systems that have untyped nodes are HyperCard and ToolBook. A typed node is labeled, and the descriptor help users determine the nature of information contained in the node. Types helps to classify nodes or define specialized operations. For example, a node of type, "video" informs the viewer the type of application required to play it. Examples of systems that have typed nodes are World Wide Web[Berners Lee 1992] and Notecards[Halasz 1989]. In the previous section, we have shown how to extend the notion of typed node with the SGML document type definition(DTD). The SGML DTD enables us to create new document types, whose instances can be linked together to form a hypermedia network.

There are several problems with the simple node-link model. First, the user can use the entire document node at only one level. Despite the elaborate hierarchy provided by the document structure, there is no way to zoom in and out of the individual document structure, examining its contents at different levels of detail. This capability is commonly found in structured editors and outline processors, and is a critical component in many authoring and information organization tasks. Another problem is the lack of mechanisms for controlling the complexity of hypertext structures. What may start out as a well structured information system can quickly become a tangled web of links and cross-references. In fact, Halasz(1989) criticized the "flatness" of structure of purely physical link-based hypertext structures, arguing that they lack a single node capturing the overall "relationship" structure. In other word, the basic node-link model lacks a composition mechanism, i.e., a way of representing and dealing with groups of nodes and links as unique entities separate from their components. It therefore clearly lacks the features for expressiveness and semantic richness. To solve these problems and augment the data model for modeling the behavior of documents, we add the notion of composite structures(composition) as a primitive construct in the basic data model. Composites can be used for modeling knowledge about documents through basic data abstraction techniques of classification, generalization, aggregation, and association. Composite structures use the specialization construct(IS_A) and the composition construct(HAS_A) to build structures which are more abstract than the actual document instances. As a specialization construct, composites support document abstraction at various levels using Classification and Generalization hierarchies. As a composition constract, composites support the conceptual (aggregation) and spatial(association) relationships between various document component entites.

4.1 Formal Model of Distributed HyperDocument

Before proceeding further, we provide a formal mode of a distributed hyperdocument. A hyperdocument as a whole may be modeled as a hyperlinked structure of Content Objects(CO)(or nodes). This view is illustrated in Figure 5. Each Content Object is an information conveying element of the hyperdocument, typically such information being presented to and perceived/interpreted by the human operator "in" the User Interface(UI) of the application. We use the term, Content Object, to refer to any "document" in the usual word

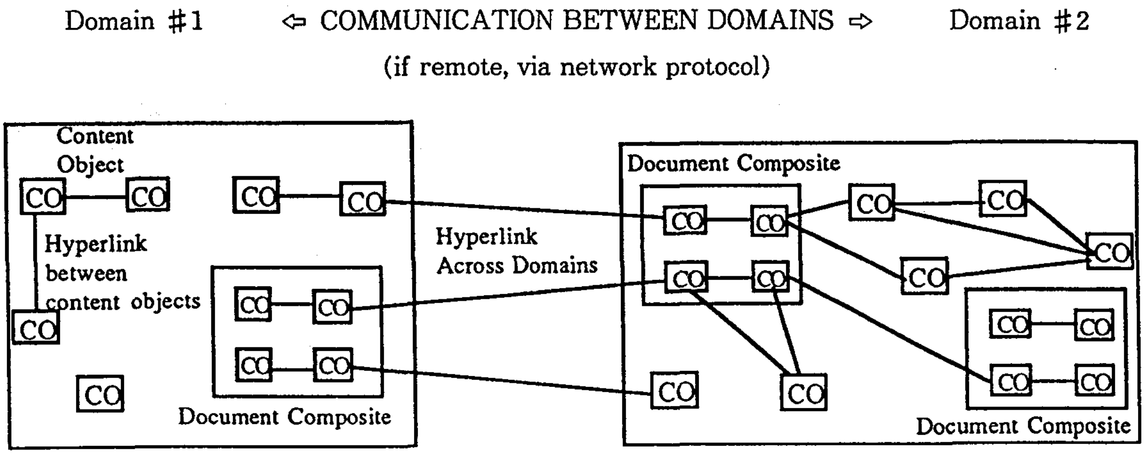


Figure 5 : Distributed Hyperdocument as Hyperlinked Content Objects

processing sense, SGML encoded document or any other multimedia file as shown in the document containers in Figure 5. Within such a document there may be "contextual links" that provide hierarchical and sequential structuring of the COs that make up the document;if a hyperlink to such a document is activated then there may be viewing/imaging steps automatically invoked by the contextual links, such as moving from page to page in a laid out(final form) document.

Multiple CO formats (data structures) are allowed to "interface"(be linked) to the hyperdocument. Each content object type must provide its own methods for formatting (content layout), viewing, editing, placing and deleting link ends. Hyperlinks are allowed between objects as long as the rules of data interchange/access are followed and the links conform to the semantics of the standardized methods defined for content objects. For instance, links can be placed from the World Wide Web documents to Gopher objects or FTP objects. Content Objects may be linked as a whole object or at addressable points within the substructure of the content(e.g., at a particular word within a paragraph). The requirement for "location or addressing models"for each content object format is the responsibility of the defining standard(and/or reference implementation) for that format. As a practical matter the "open protocol" must have a way of identifying the points to be linked to (link ends or anchors). Requirements for hyperlinking and location models are thoroughly explored in the HTML[Berners-Lee and Connolly 1994] and HyTime[ISO-10744 1992] standard. The HTML and HyTime standard specify how certain concepts common to all hypermedia documents can be represented using SGML. These concepts include : association of objects within documents with hyperlinks, placement and interrelation of objects in coordinate space and time, and inclusion of non-textual data in the document. An "object"(similar to our Content Object) in HTML and HyTime are part of a document, and is unrestricted in form-it may be video, audio, text, a program, graphics, etc. However, HyTime does not specify how document objects are encoded or interpreted by computer programs. But by using standardized linking, alignment, and addressing methods, it ensures that those objects are made available to programs in a standardized way.

Content Objects are categorized as either Processable(PCO) or Formatted(FCO). The PCO is an editable, viewable or computable form. It is a format supporting further revision and evaluation in authoring/editing and dynamic information processing environments (such as software programs, content based retrieval, linguistic analysis, spreadsheeting, database accesses, modeling, etc.). The FCO is ready for presentation, display or imaging. Typically it is derived from the PCO for a particular imaging device (viewer/player)by formatting or otherwise transforming the PCO(e.g., plotting a graph from tabular numeric data, accessing records from a database by evaluating an SQL query and formatting them for viewing, etc.).

4.2 Aggregation For Composite Documents

Documents may be treated as Composite Content Objects using aggregation. Aggregation is an abstraction operation that clusters related component document (or document objects) and forms a higher-level object. In other words, aggregation is the operation of constructing more complex phenomena out of a basic set of component documents. The relationship between the lower-order object and the higher order object is described as "IS_PART_OF" relationship. Note that we use the term objects rather than documents to imply a difference in the level of granularity. For example, a compound document such as a company's annual report, is an aggregation of component document instances such as charts, graphs, tables, spreadsheets and textual paragraphs. Aggregation supports the notion of inclusion. Inclusion can be implemented within the hypermedia network itself as opposed to a layer on top of the network. For example, a compound document can be a part of a document network that is referenced by other documents rather than being a separate entity. Moreover, all aspects of hypermedia (audio, video and im ages) should support inclusion (or IS_PART_OF) relations as a construct distinct from standard(reference) links. Whether or not inclusion relations share a common implementation mechanism with standard links is unimportant, so long as the semantics of inclusion, as opposed to reference, are fully supported. Both processable (PCO) and formatted (FCO) Content Object can be included as component Content Object in the hyperdocument; it is thus possible for both the PCO and the FCO of the same object to co-exist in the same hyperdocument(e.g., a Postscript formatted document derived from an SGML/HyTime source) and be linked to avoid time consuming derivations such as formatting or other compute-intensive content transforms(e.g. changing color model, decompressing an image format).

4.3 Document Association

Association is the assignment of document instances and aggregated entities to sets, using criteria different from those used for classification. Association is useful as hypermedia systems tend to have difficulty with rapidly changing information. This difficulty arises from the essentially static nature of the hypermedia data model which encodes information into a collection of independent nodes interconnected into a static network. This network does not change unless it is explicitly edited by the user or some other external agent. In particular, the network cannot reconfigure itself in response to changes in the information it contains. This lack of dynamic mechanisms limits the utility of hypermedia in many task domains. Knowledge about the critical dimensions of the document space, the characteristics which distinguish one document from another, and appropriate naming schemes develops over time as the users become familiar with the information. A problem arises because the referencing or linking tasks all require the user to have such knowledge up front. As the knowledge of the information space evolves, previous organizational commitments or structures become obsolete.

Association allows the definition of new names for a collection of document instances. There are many ways the associations between documents may be captured; as binary associations of entities, as n-way associations, as physical links between two entities or as logical links. A physical link is one where one document component contains a reference that denotes the location of another document component in storage. A logical link is where one document component contains a value that identifies another document component by one of its properties. It is the latter that we are interested in. These names are in efect "persistent entities" that endure within the system from one session to another. These named entities serve as the starting points in the query expressions.

5. DOCUMENT DATABASE NAVIGATION AND BROWSING

A major premise of the work described in this paper is that embedded knowledge provided by descriptive markup can be very important for document retrieval, query processing and the design of user-document interfaces. This extra knowledge provided by descriptive markup allows us to design new and sophisticated manipulation mechanisms that conventional query and retrieval systems cannot match. Our goal in this paper is to bring about an integration of information retrieval and database systems in designing retrieval tools for unstructured or hybrid organizational data. In this section, we first describe the challenges in integrating document network navigation with query mechanisms or the "Query by Browsing" paradigm. We then examine how structure-based search and retrieval is made possible by descriptive markup.

Any large database of structured documents will be accessed by casual or naive users who are not experts in the subject of the documents. Such users will typically want to browse the database to locate items in which they might be interested. The design of user interfaces for document databases is an area in need of more attention as existing user interfaces are often unfriendly (text based)or difficult for non-experts to use. In this section, we outline the design of a "Query by Browsing" user-interface paradigm, called HyBrow, that is appropriate for "interactive browsing" through large SGML encoded documents (such as technical manuals) and interlinked document networks. We wish to avoid the traditional "type a query of key words or phrases" interface (or content searches) popular in information retrieval environments and textual query languages. As a result, we are focusing on a "move and zoom" interface that allows a user to "navigate" around in an N dimensional hyperdocument space with little or no typing. The main reason for the "move and zoom" interface is to allow the user to understand the inter-document hypertext structure and avoid the problem of being lost in hyperspace [Nielsen 1990]. This requires the development of sophisticated document network visualization techniques coupled with query and browsing methods for documents (see Figure 6 below). Note by document, we mean both individual content object and the network of content objects.

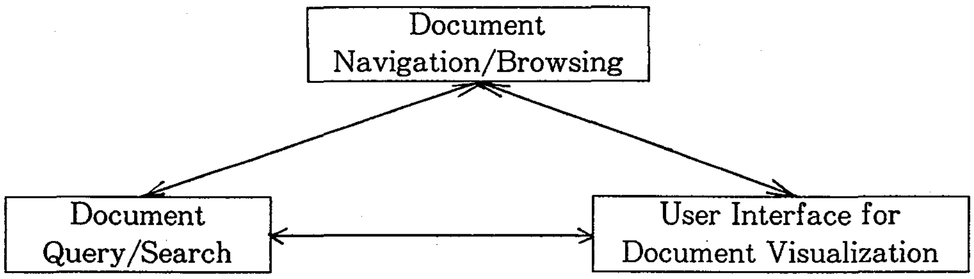

Document Navigation/Browsing
Document Query/Search
User Interface for Document Visualization

Figure 6 : Different Facets of Document Manipulation

For the sake of understanding, we separate the docunent manipulation problem into two areas : Document Network "Query By Browsing"and Document Content "Query By Browsing".

5.1 Document Network "Query By Browsing"

Our approach is unique as we attempt to integrate/interleave document network search/query and document network navigation or browsing. This interleaving of query and browsing takes place because they are inter-related activities. Their relationship is essentially this : a user will browse through a large document network, identify the region in the document network which seems to have documents he is inthrested in. The user then queries this region(or subject area) to further focus or narrow the search area. The task of querying is analogous to finding/extracting subtrees of the document network structure, and mapping one structure to another. Here, we are making the assumption that in interdocument networks it is possible to differentiate between organizational links (see Figure 7b) and referential links (see Figure 7a). Organizational links are important for imposing a logical structure on the hypertext. Systems that have purely referential links can quickly turn into a chaotic mess. A referential link is a pointer from one document component (Content Object) to another. These document components may be contained in the same compound document. For example, the World Wide Web (which has only referential links)is an information system containing enormous amounts of valuable information which cannot be retrieved since there is no structure present. In this environment, users either have to know the exact location of the document or apply a "hit and run"method of traversing the links and making intelligent guess at each node to get to the document of interest. Needless to say this is quite time-consuming and inefficient.

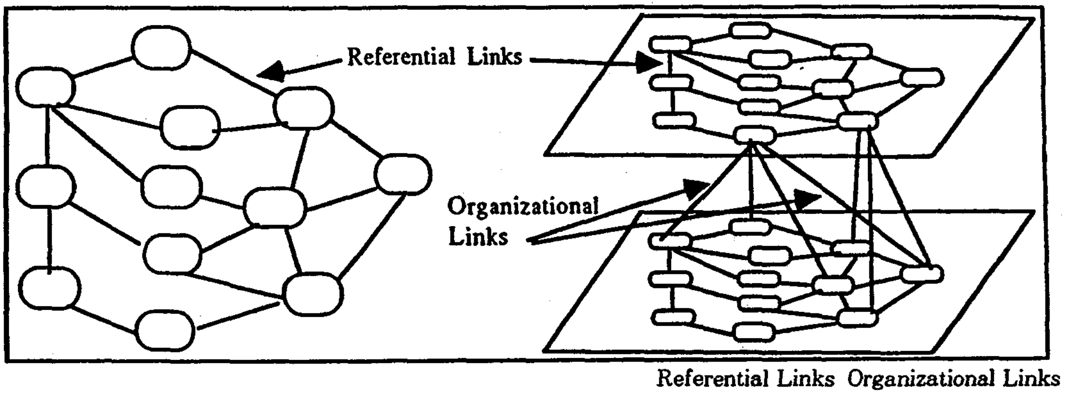


Figure 7a : Referential Hyperlinks

Figure 7b : Organixational Hyperlinks

Organizational links are used to create a hierarchy while referential links are used to cross - reference information. Since organizational structure reflects certain semantic infomation, it is quite possible to have many co-existing hierarchical structures on the same body of information. This is quite feasible with the notion of composite structures(especially with association structures described in section 4). We can imagine multi-dimensional hypermedia structures consisting of links between composite structures providing a higher level of abstraction. Each composite structure created by document association can be thought of as a predefined category based on certain attributes of underlying content objects which are at a lower level of abstraction. The user navigates through the organizational hierarchy of the document database by essentially traversing the lattice structure that is created by these document associations. Once the user has selected an individual(coule be extremely large(e.g. book), then the user would have to use the logical structure of the document to repeat the process of browsing and querying.

5.2 Document Content "Query By Browsing"

In order to browse through structured documents, it is important to access purely stuctural information about the document instance being looked at. The structural information is analogous to a table of contents for a book. Structural information is provided through the descriptive markup. Descriptive markup is used for specifying the logical components of documents such as chapters, headings, paragraphs, figures, annotations etc. and the hierarchical relationships among these components within the documents. This has obvious implications for the browsing and viewing. The logical structures supported by descriptive markup can be very sophisticated. In the case of a SGML document the structural information is available through the so-called Element

Structure Information Set(ESIS). The implementation structure of a "query-by-browsing"architecture of SGML document content is shown in Figure 8. From the figrue, we can see that the event stream that is flowing from the SGMLS parser[4] to the ESIS interpreter is the key for document browsing.

First let us examine how the system works. First, the user selects a document for browsing. On receiving the user's selection the SGML document database send the selected document instance to the SGMLS parser. The parser reads through the character stream of the document instance and interprets the characters according to the SGML declaration and the Document Type Definition. The result of this interpretation is a parse tree of the document instance. The parse tree contains information about the elements of the document instance structure and their attributes. This parse tree is sent to the ESIS interpreter for further processing as an event stream. To make this clear look at the following SGML document instance which is transformed into a parse tree.

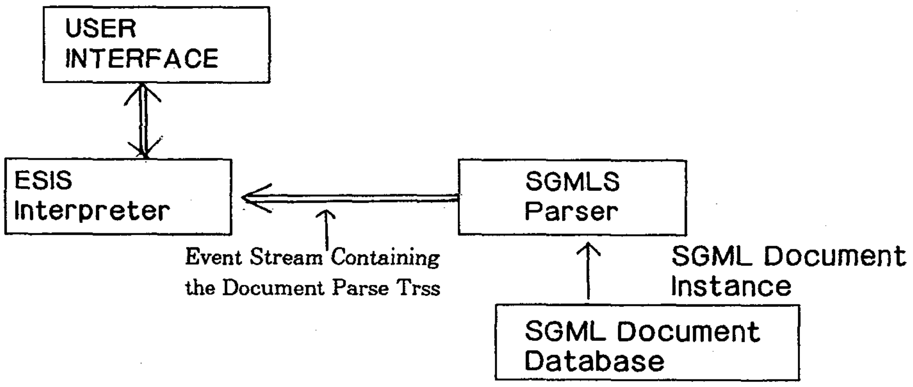


Figure 8 : Query-by-Browsing Implementation Architecture for a Document Content

```
<!DOCTYPE memo[
<!ELEMENT memo -- (subject, body)>
<!ELEMENT subject -- #PCDATA>
<!ELEMENT body -- (para+)>
<!ELEMENT para - O#PCDATA>
]>
<memo>
<subject> Material Shortage on Production Line 2</
  subject>
<body>
<para> We are short of 50 PCB for satisfying custom-
  er orde number #50089.
<para> Please expedite the PCB order from supplier.
</body>
</memo>
```

The parse tree from the SGML parser will be :

```
MEMO START
SUBJECT START
#PCDATA "Material Shortage on Production Line2"
SUBJECT   END
BODY    START
PARA    START
#PCDATA "We are short of 50 PCB for satisfying
  customer order number #50089."
PARA    END
PARA    START
#PCDATA "Please expedite the PCB order from
  supplier."
PARA    END
BODY    END
MEMO    END
```

4. SGMLS is a free public-domain parser written by James Clark for parsing SGML documents. It is available from(ftp : //ftp.ifi.uio.no/pub/SGML/).

Note that even though there are no end tags on the PARA elements in the SGML document instance, they are inferred by the parser and end events are generated accordingly. This parse tree is sent via the event stream to the ESIS interpreter. The sequence of information is as if doing in-order traversal of the parse tree. At each node in the parse tree information is given to the ESIS interpreter about the name of node (called the elements's Generic Identifier-GI) and information about its attributes. Depending on the user's request for browsing or querying the ESIS interpreter would respond appropriately and send the desired result to the user interface. The user interface would then format the results using a Format Specification language--FOSI[MIL-M-28001A) or DSSSL Adler 1994] and present the results accordingly.

Conclusions

To summarize, the development of a coordinated electronic document management strategy for accessing and manipulation structured multimedia documents involves several issues that are consistent across applications. These issues can be classified into the following categories :

- Data Model View--responsible for structuring the storage of entire documents, parts of documents and the hyperlinks between documents;
- User Interface--responsible for communication and interaction with end-users. This involves the movement of multimedia documents over computer networks, and display and manipulation of documents;
- Model-Interface Translators--which synchronize the interface events(content or structure based queries)with the data model view.

In this paper, we have addressed the following questions that were derved from the above mentioned issues. First, how to encode or represent the logical structure of documents.

To answer this question, a data model for internal content representation using SGML was presented. We then addressed the question of organizing and structuring the encoded documents into larger useful patterns using the notion of composite structures. We have presented frameworks for capturing document behavior using the abstraction techniques of document aggregation and association. Finally, the critical issues concerning user interface design for document collections were discussed. We presented a new paradigm called "Query by Browsig" that integrates the querying about document networks, with navigating hyper document links to find information of preference. The implementation strategy for "Query by Browsing" individual document content was discussed. Clearly, the strength of markup languages in supporting information reuse, cross-platform reformatting, intelligent navigation, and executable interactive documents, will make it an integral part of the future of database management systems. As we move toward storing greater amounts of information in documents rather than in relational databases, research that was presented in this paper will be useful in designing better information systems.

References

Adler, Sharon C. "DSSL-document Style Semantics and Specification Language", *ISO/IEC DIS 10179 : 1994. Text Composition-Document Style Semantics and Specification Language(SAAL)*, 1994.

Akscyn, R., D.L. McCracken, and E. A. Yoder, "KMS : A distribyted hypertext for managing knowledge in organixations", *Commun. of the ACM*, Vol. 31, No. v, 7 July 1988, pp. 820-835.

Arnon, Dennis, "Scrimshaw : A Language for Document Queries and Transformations", *Proc. Electronic Publishing 94 Conference* (April 13-15, 1994, Darmstadt, Germany), John Wiley and Sons, 1994.

Berners-Lee, B., R. Cailliau, J. Groff, and B.Pollermann, "World-Wide Web : The Information Universe", *CERN Technical Report (the European Laboratory for Particle Physics)*, Geneva, 1992.

Bertino, Elisa, Rabitti, Fausto, and Gibbs, and Simon, "Query Processing in a Multimedia Document System," *ACM Transactions on Office Systems*, Vol. 6, No. 1, January 1988, pp.1-41.

Botafogo, R.A. and B. Shneiderman, "Identifying aggregates in hypertext structures", *Proceedings of Hypertext 91*, ACM, New York, Dec 1991, pp.63-74.

Bryan, Martin, *SGML : An Author's Guide to the Standard Generalized Generalized Markup Language*, Wokingham/Reading /New York : Addison-Wesley, 1988.

Chomsky, Noam, "Three Models for the Description of Language", *IRE Transactions on Information Theory*, Vol. 2, No. 3, 1956. pp. 113 : 124.

Conklin, J., "Hypertext : an introduction and survey", *IEEE Computer*, Vol. 20, No. 9, 1987.

Coombs, James, Allen, Renear, Steven J. Derose, "Markup Systems and the Future of Scholarly Text Processing", *Commun. of the ACM*, Vol. 30, No. 11, 1987. pp. 933-947.

Croft, B. and R. Thompson, "13R : A New Approach to the Design of Document Retrieval Systems," *Journal of the American Society for Information Science*, Nov 1987.

Donzeau-Gouge, V. and G. Kahn, B. Lang, and B.Melese, "Document Structure and Modularity in Mentor", *SIGPlan Notices*

Vol. 19, No. 5, May 1984.

Furuta, R., "Concepts and Models for Structured Documents", *Structured Documents*, Cambridge Series on Electronic Publishing 1987, pp. 9-27.

Goldfarb, Charles F., *The SGML Handbook*, Edited and with a foreword by Yuri Rubinsky, Oxford : Oxford University Press, 1990.

Goodman, D. "*The Complete Hyper Card Handbook*". Bantam Books, New York, 1986.

Halasz, F. and M. Schwartz, "The Dexter hypertext reference model", K. Gronbaek and R.Trigg, Eds.,*Commun. of the ACM* Vol. 37, No. 2, Feb 1994.

Halasz, F.G. "Reflections on NoteCards : Seven issues for the next generation of hypermedia systems", *Commun. of the ACM* Vol. 31, No. 7. July 1988, pp. 836-855.

Haviland W., "SGML Frees Information", *Byte*, June, Vol. 17, No. 6, 1992, pp. 279.

Herwijnen, Eric van, *Practical SGML*. dordrecht / Hingham, MA : Wolters Kluwer Academic Publishers, 1990.

ISO-8879, *Information Processing-Text and Office Systems-Standard Generalized Markup Language(SGML). Intermational Organization for Standardization*, Ref. No. ISO 8879 : 1986(E). Geneva/New York, 1986.

ISO-number 8893, *Information Processing-Text and Office Systems-Open Document Architecture(ODA). International Organization for Standardization*, Ref. No. ISO 8879 : 1986(E). Geneva/New York, 1986.

ISO/IEC-10744, *Information Technology-Hypermedia/Time-based Structuring Language(HyTime). International Organization for Standardization*, Ref. No. ISO 8879 : 1986(E). Geneva/New York, 1992.

Kahle, B., *Wide Area Information Systms (WAIS)*, Technical Report, ThinKing Corp. Boston, 1991.

Kimura, G., *A Structure Editor and Model for Abstract Objects*, Unpublished Doctoral Dissertation, Unversity of Washington at Seattle, July 1984.

Knuth, Donald, "The Errors of TEX", *Software-Practice and Experience*, Vol. 19, No.7, July 1989, pp. 607-685.

Lamport, L., "Document Production : Visual or Logical", *Notices of American Mathematical Society*, Vol. 34, 1987.

Lamport, L., *LaTEX : A Document Preparation System*, Addison-Wesley, Reading, Mass., 1986.

MacLeod, Ian, "A Query Langauge for Retrieving Information from Hierarchic text Structures", *Compueer Journal*, Vol. 34, No. 32 1991, pp. 254-264.

MIL-M-28001A : *US Department of Defense, Military Specification, Markup Requirements and Generic Style Specifica-*

tion for Electronic Printed Output and Exchange of Text(SGML), CALS Phase 1, 1 Core Requirement Document. MIL-M-28001A Draft(Superseding MIL-M-28001, 15 December 1988). 17 July 1989.

Naur, P. et al., "Report on the Algorithmic Language, ALGOL 60", *Commun. of ACM* Vol. 3, No. 5, 1960, pp. 299-314.

Nielsen, J. "The art of navigating through hypertext," *Commun. ACM*,Vol. 33, No. 3, 1990, pp. 296-320.

Notkin, D., *Interactive Structure-Oriented Computing*, PhD Thesis, Carnegie Mellon University, Department of Computer Science, CMU-CS-84-103, 1984.

OpenDoc Technical Summary, *Component Integration Laboratory*, Apple Computer Inc., 1993.

Peels, A., N. Janssen and W. Nawijn, "Document Architecture and Text Formatting", *ACM Transactions on Office Information Systems* Vol. 3, No. 4, 1981.

Raman, T., *Audio System for Technical Readings*, Unpublished Ph.D. Dissertation, Cornell University, Ithaca, New York, 1994.

Reps, Thomas, and T. Teitelbaum, *The Synthesizer Cenerator : A System for Constructing Language-Based Editors*, Springer-Verlag(Texts and Monographs in Computer Science), New York, 1989.

Salton, G. "*Developments in automatic text retrieval*". Science Vol. 253 : 5023, Aug 30, 1991, pp. 974-980.

Stonebraker, M. and G. Kemnitz, "The POSTGRES Next-Generation Database Management System", *Commun. of ACM*, Vol. 34, No. 10, 1991, pp. 78-92.