The GILBANE REPORT™

on Open Information & Document Systems

Vol. 1, No. 2
May 1993

Publisher:
Publishing Technology Management, Inc.
Applelink: PTM

Editor:
Frank Gilbane
fgilbane@world.std.com
(617) 643-8855

Subscriptions:
Carolyn Fine
carolyn@world.std.com
(617) 643-8855

Design & Production:
Catherine Maccora
(617) 241-7816

Associate Editor:
Chip Canty
ccanty@world.std.com
(617) 265-6263

Contributing Editor:
Rebecca Hansen
MCI Mail: 4724078

SGML & SGML Open

We have been asked to explain what SGML is countless times in the last few years. In the past year however, the inquiries have increased dramatically, and come more from senior business managers and industry analysts than technical managers — a sure sign that the standard is moving into the mainstream.

The recent launch of the consortium of SGML suppliers, SGML Open, and the growing press coverage resulting from it has further increased the demand for information about SGML. Accordingly, we decided to write a non-technical description of the role SGML plays in information systems as an introduction to a discussion of the new consortium.

Document Query Languages

Most information system managers, and many system users, are familiar with the methods and issues involved in extracting information from databases. The use of SQL for querying relational databases is widespread, and whether hidden behind a graphical user interface or not, SQL is well understood.

Most computer users are also familiar with some form of text retrieval, whether a simple find command associated with a word processor, or more specialized keyword or full-text based applications.

As more information is being created and maintained in structured form, interest has grown in using structural information to enhance information retrieval capabilities.

Managers contemplating managing large document databases, or who want to access information that may be in either a database or document system, need to choose which combination of these approaches makes sense for them. Our second article reviews the issues that need to be considered.

Contents

SGML Open — Why SGML & Why A Consortium?

An ISO standard since 1986, SGML (Standard Generalized Markup Language) is arguably the most critical information encoding standard for open document systems. Although SGML products have been available for years, the past twelve months have seen an explosion in interest in SGML, and vendors have scrambled to answer this interest with new products. Both trends recently culminated in a new industry consortium that brings SGML users and vendors together: SGML Open.

Like other consortia, SGML Open provides a window onto the political agendas of competing vendors (and, in this case, standards organizations). While we will not cover political activity, we will examine a few legitimate concerns raised by some vendors, users, and standards developers. As always, our goal is to extract from the debate the critical information that you, as IS managers, need to make technology and business decisions.

To lay the groundwork for our discussion of the purpose of SGML Open and what it means to you, this article begins with an "executive overview" of the business problems SGML helps to solve in information systems.

SGML — Executive Overview

SGML is a powerful and complex standard, with many subtleties that make it difficult to describe even to persons with a technical background. We will not attempt a detailed description here, but will describe the business problems SGML addresses, provide an accessible, conceptual description of how it works, and look at the driving forces behind the acceleration of SGML activity. For those interested in further technical detail, there is a growing collection of literature and tutorials and seminars.[1]

"Most succinctly, the business problem SGML addresses is that of managing strategic information."

The Business Problem

Most succinctly, the business problem SGML addresses is that of managing strategic information. Corporations have certain types of strategic information (*e.g.*, customer lists, proprietary research) in database form, but the majority of corporate information is in documents, whether on paper or in electronic files.

Strategic information for manufacturing companies includes product design, logistics and support information; for pharmaceutical companies it includes experimental and clinical test results. Insurance, utility, petrochemical and other companies each have other kinds of critical information in document form.

What makes this information strategic? The answer is found by measuring its contribution to the success of the enterprise. The value of strategic information can be established (at least approximately) by asking a few questions. How much would it cost to replace? How long would it take to replace? What percentage of corporate revenues are directly related to the *information* portion of your product?

An Example

Consider, for example, a manufacturer of heating and cooling subsystems for the transportation industry, the hypothetical Global X Corporation. Its products are used on planes, trains, cars and boats; they are sold to manufacturers of these products and to

1. The Graphic Communication Association (GCA) has the most complete list of books & articles related to SGML. The GCA provides seminars and tutorials and maintain a list consultants and vendors who provide similar services. They can be reached in Alexandria VA at (703) 519-8160. Also, see <*Tag*>, a newsletter devoted to SGML, and available from SGML Associates in Denver, CO.

those who use and maintain them. Global X buys components for its systems from many different suppliers. Important specifications about Global X's products (such as the threshold for a safety cutoff switch) must be shared with suppliers to ensure that every component that Global X purchases will work. Global X must also work within specifications set by its own manufacturing customers such as for the required level of electrical shielding for wire. (See Figure 1)

Much of this information must be available in both database form (*e.g.*, for inventory control) and in document form (for maintenance manuals). Furthermore, the integrity of the information is critical to both the safety of the operator and quality of the product.

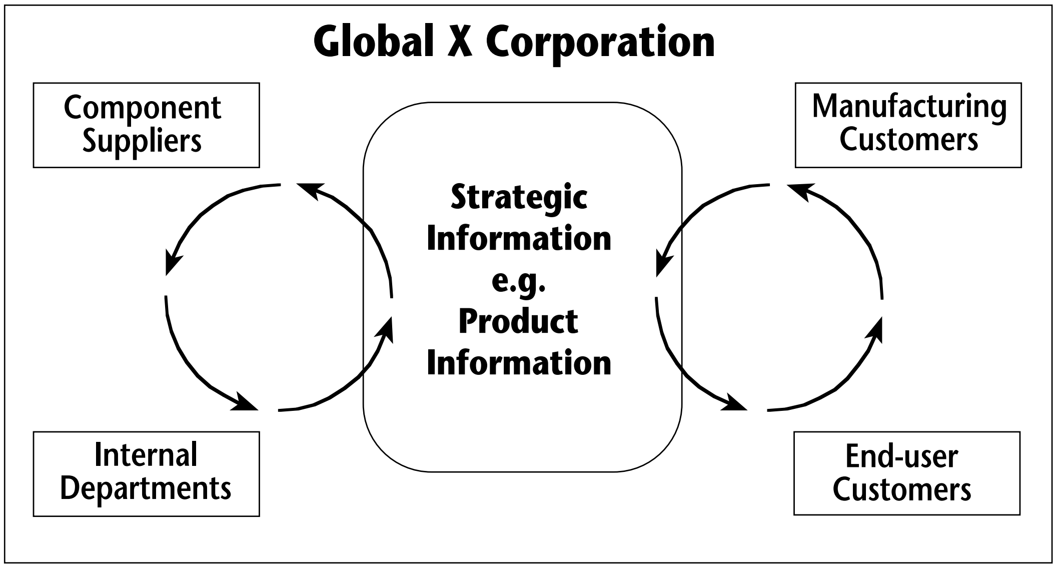


Fig. 1 Integrating and Sharing Strategic Information

Determining how to share this kind of information — in a cost-effective way — is a huge challenge facing companies today. Costs to consider include those of sharing information with suppliers, with customers and with other departments, as well as the cost of migrating applications and data to new computer equipment and software applications over time.

SGML helps alleviate these problems — how?

What Does SGML Do?

SGML was originally developed to allow publishing systems to exchange information in document form. The developers of the standard were publishing systems users.[2] When they started analyzing documents to see what was making interchange so difficult, they recognized three types of information maintained for every document (see Figure 2):

- its data content (what the document says),
- its presentation or format (how the document looks), and
- its logical structure (how information is organized within the document).

The authors of the standard saw that these three types of information are often directly related to each other — the more important the heading, the larger the type. But they also saw that this was not always true — one cannot assume that every instance of a phrase in 10 point bold is a figure title. It became clear that every system could interpret

2. Although some of these developers, like the editor, Charles Goldfarb, worked for vendors, they mostly worked for in-house publishing organizations rather than product development organizations

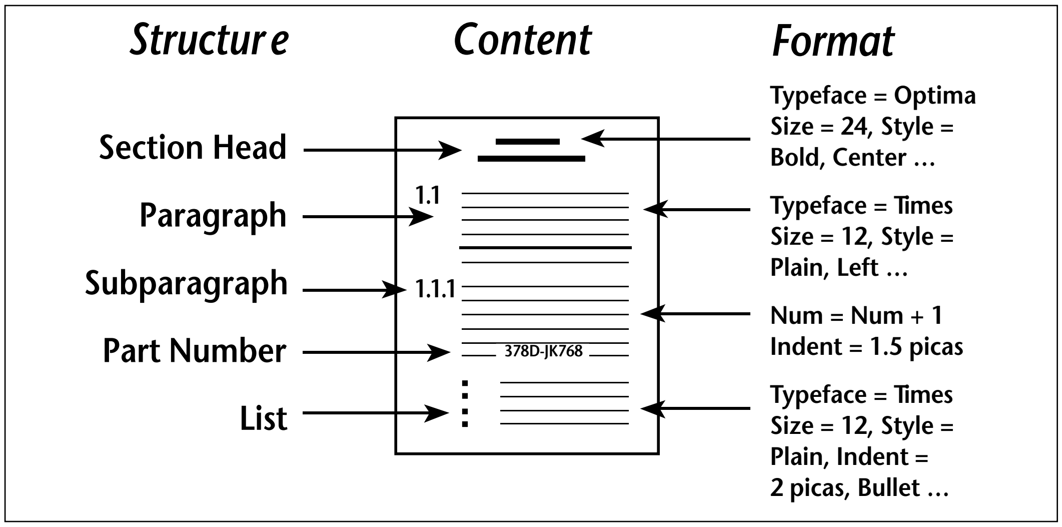

Structure
Content
Format
Section Head
Paragraph
Subparagraph
Part Number
List
1.1
1.1.1
378D-JK768
Typeface = Optima Size = 24, Style = Bold, Center ...
Typeface = Times Size = 12, Style = Plain, Left ...
Num = Num + 1 Indent = 1.5 picas
Typeface = Times Size = 12, Style = Plain, Indent = 2 picas, Bullet ...

Fig. 2 The Three Types of Information In Documents

and represent the content of documents, but that their structural information was mostly ignored (except to the extent it was reflected in the way the documents were formatted). On the other hand, how formatting information was dealt with often differed widely from one system to another.

Then and now, the real problem for interchange is that each publishing system uses a different language to identify parts of documents and to attach formatting characteristics to them. Not only do these languages differ, but they reflect capabilities of different proprietary products.

"There are many companies that want to reuse information for more than exchanging documents between publishing

For example, System A may have one or more commands in its language for adjusting the amount of space used to indent the first line of a paragraph (as in the subparagraph in Figure 2). System B might have the same *capability*, but employ an entirely different *method* to adjust this space (for example, it may make a relative adjustment from surrounding text elements instead of adjusting from the absolute left margin). System C may not be able to make this kind of adjustment at all.

SGML deals with this problem by providing a standard and robust, yet rigorous, way to interchange document *content* and *structure*, leaving the issue of how to format the information to individual applications. This is why you hear about the importance of separating structure and content from *form*. After all, the reasoning goes, isn't it the data content that is important? And isn't it the organization (structure) of the content that provides us with a context for correctly interpreting it?

Form, on the other hand, is a by-product of a particular application. Not only is the formatting information the most difficult to share, it is also the least important.[3]

Since it is the formatting features that differentiate their products, vendors at first had no interest in a common formatting language. They were, however, at least willing to consider a standard, like SGML, that dealt with content and structure only.

Why Is SGML Of Interest To IS Managers Today?

Today we expect more from SGML. There are many companies that want to reuse information for more than exchanging documents between publishing systems. We want to be able to "publish" the same information on different media, and we want to engineer and manage document information to improve our business processes.

3. This is not to say that formatting information is not useful and important — just less so for certain applications. There are cases where the formatting information is *at least* as important as the content, e.g., advertising. We will cover the challenge of interchanging formatting information in a future issue.

Indeed, if we design our information appropriately, it can be reused in powerful ways — *information engineering* does not (or should not) refer only to information in databases. Information should be re-engineered so that it can be used in both databases *and* documents.

In our previous issue we discussed an issue too often ignored in the open systems debate: the importance of *open information* as opposed to open platforms and open networks. Figure 3 illustrates how open information can be used by different applications, as well as by different vendors addressing the same application. In this example, it is inefficient and costly to have the same *part number* object replicated in each of several applications that need to use it; instead we want to store the part number once and allow each application to share it.

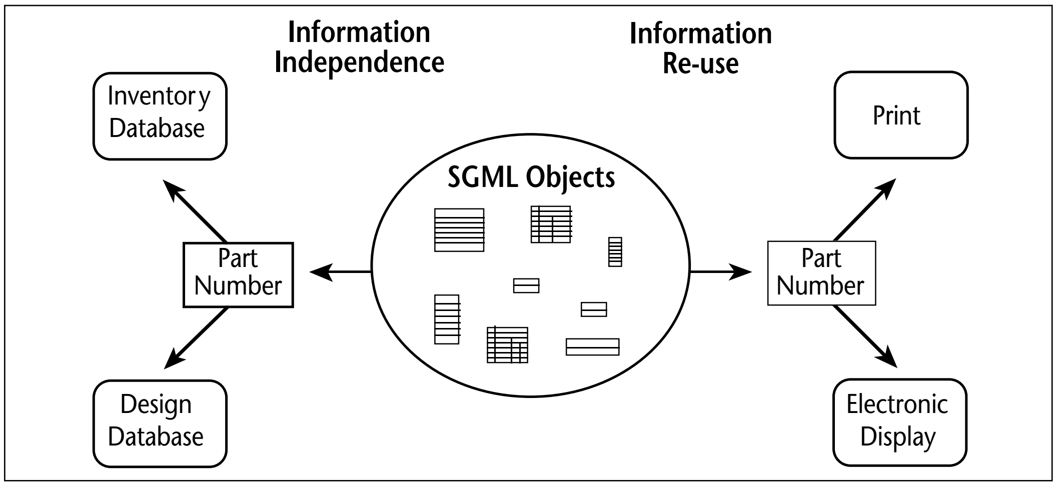


Fig. 3 Application Neutral Information Bases

Relation To Client/Server

Figure 3 also illustrates the role of SGML in client/server environments, where multiple applications (clients) can access shared information from a server. The more heterogeneous your client/server environment, the more value SGML can provide. It is especially important to what Apple now calls *client/client/server* applications, where information must flow freely from a server to a desktop to a handheld personal digital assistant; in such environments, file-system independence is even more critical.

Three applications that lend themselves well to the client/server model are document, data and image management. SGML is especially well-suited to such applications, because it facilitates the sharing of document information elements (including images[4]) residing on the data server. Many different client applications can retrieve and update SGML-tagged information from such server-based repositories.

The Driving Forces Behind SGML Adoption

Not every good idea gets taken seriously, and the best products do not always make it in the marketplace. Whether or not we agree that SGML is a good idea, if it does not have enough market momentum behind it we may not feel safe incorporating it into our information-management strategies.

SGML today is showing such momentum, in at least four ways:

It provides real value to users. SGML protects investment in information because it is vendor independent, it reduces information management costs by allowing data to be re-used rather than re-created, and it contributes to greater interoperability between multiple document system applications.

4. . AIIM (the Association for Information and Image Management) and its members are developing an SGML application to facilitate interchange of information about images.

It has strong vendor support. All major document system vendors offer products that support SGML today.

It supports future technology. SGML is not a standard that will hold back technology. It can even be used to describe complex, object-oriented, multimedia databases.

It is being used successfully to define industry requirements. Large vertical industries (aerospace, automotive, commercial aviation, computer, defense, electronics, telecommunications) as well as government agencies have begun to require support for common SGML applications.

Vendors can support a standard either by competing with one another to implement particular applications, or by cooperating to ensure that all such applications work well together to solve larger problems. In the case of SGML, vendors are now doing both.

SGML OPEN

After its first meeting in February 1993 at the at the Winter TechDoc conference in San Antonio, Texas, the SGML Open consortium was officially launched in April at Seybold Seminars in Boston. Its 33 founding members represent virtually all the current major suppliers of SGML solutions, as well as a few large organizations that are primarily SGML users.

Elections for officers will be held in June. In the meantime, the consortium's acting executive committee consists of Yuri Rubinsky, chairman; Larry Bohn, president; and Haviland Wright, chief technologist. (Rubinsky is president of SoftQuad, Bohn is senior VP of Interleaf, and Wright is president of Avalanche Development Company.) The Board of Industry Advisors includes Esther Dyson, publisher of *Release 1.0*; Frank Gilbane, editor and publisher of this report; Charles Goldfarb, editor of the SGML standard; and Jonathan Seybold, president of Seybold Seminars and publisher of Seybold Publications.

"If SGML is already an accepted international standard, then why do we need a vendor consortium?"

Relationships with the Graphic Communication Association (GCA), the GCA Research Institute (GCARI) and the international SGML Users Group were also announced. See Figure 5 for the official organizational structure.

According to its mission statement "SGML Open is a non-profit, international consortium of providers of products and services, dedicated to accelerating the further adoption, application and implementation of the Standard Generalized Markup Language, the international standard for open interchange of documents and structured information objects." The consortium intends to accomplish this by focusing on:

- marketing the benefits of SGML to commercial industry, and
- working together to increase the level of interoperability between different kinds of SGML applications and different SGML products.

It is worth noting that, unlike some other vendor consortiums, SGML Open was not formed to oppose or punish another vendor or group of vendors. Nearly every SGML vendor of note has already joined SGML Open; no one has been left out of the loop, intentionally or otherwise. This is particularly good news for companies interested in adopting open systems based on open information.

Do We Need A Consortium?

The most common question about SGML Open from the user community has been "if SGML is already an accepted international standard, then why do we need a vendor consortium?"

Users seem to have two different reasons for asking this question. Some fear that having vendors talk to each other may do more to compromise the standard than to implement it, *i.e.*, do more harm than good. Others wonder whether the founding of the consortium implies that something is wrong with SGML itself.

Both concerns are reasonable and should be addressed.

Marketing The Benefits Of SGML

One reason why users will benefit from a vendor consortium is that it is in each vendor's interest to educate potential adopters of SGML to understand the value of purchasing SGML solutions. SGML is difficult to describe, and vendors to date have been only partially successful in marketing it; customers seem to have trouble distinguishing the capabilities of a particular vendor's product from the features of SGML itself.

We can now expect that both the quality and consistency of vendor messages regarding SGML will improve. The consortium has explicitly disclaimed interest in policing claims of SGML compliance, but since all major SGML vendors joined the consortium, one can expect a healthy degree of self-policing.

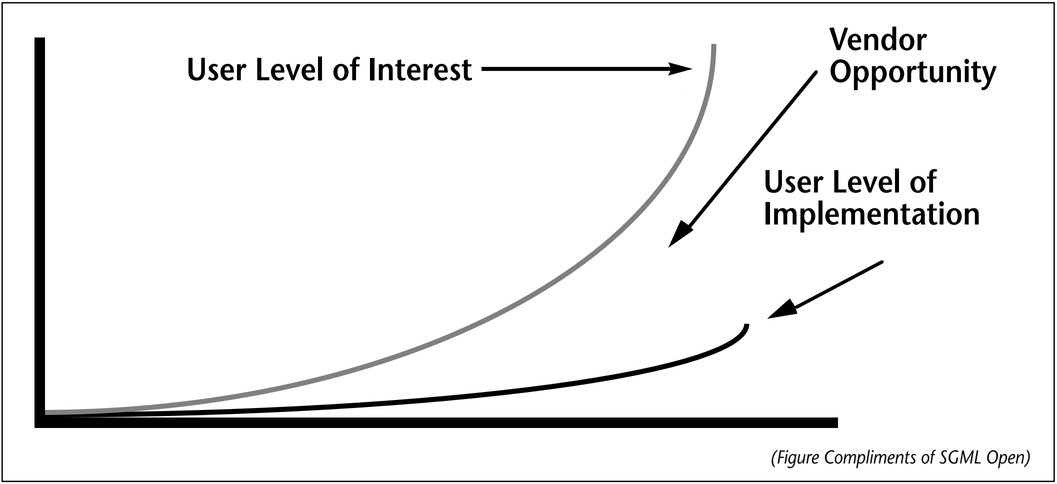


Fig. 4 SGML Interest Versus Implementation

The vendors of course want to sell their products and to recover the often significant costs of developing sophisticated SGML tools. Their collective view of the opportunity in SGML market is illustrated in Figure 4. The more successful the vendors are, the more they will invest in open information solutions based on SGML, and the more they will compete to develop SGML products that are even easier to use. We should cheer them on.

The Interoperability Issue

The second reason we need a consortium is even more compelling. SGML is a powerful enabling standard that is not limited, in any way, in the kinds of documents or information it can represent. This flexibility is a basic user requirement of the standard.

How we take advantage of this flexibility will change over time. SGML, for example, does not constrain us in any way to use a particular graphics (or math or tabular) notation, nor does it tell us how we must describe or identify each such notation. As better notations are developed, we can decide when and how to use them within SGML documents.

But for true interoperability, everyone involved must agree on what notations to use and how they are identified. In other words, the freedom SGML gives us carries with it an obligation to communicate with each other about *how* we use the standard. It is clearly in SGML's favor that it lets us use any type of data, and that it provides a well-defined mechanism for including data in other notations. But both parties to the exchange of SGML documents must share an understanding of how that mechanism will to be used.

In other words, SGML may be *necessary* for exchanging documents effectively, but it alone may not provide *sufficient* information to allow others to accept and interpret them. SGML is an enabling standard; implementing it effectively for a given purpose at a given time requires communication and, perhaps, additional standards or protocols that extend SGML in various ways.

Remember, it is precisely *because* SGML is so robust and flexible that we need to facilitate interoperability, not because it is lacking in some way. The vendors in SGML Open can contribute significantly to interoperability by helping their *users* determine what kinds of guidelines make sense for their applications, and by working together to make sure their products support these users' needs.

"Remember, it is precisely because SGML is so robust and flexible that we need to facilitate interoperability, not because it is lacking in some way."

The Issues

Other concerns have been raised by members of ANSI and ISO standards bodies. Although in our view the consortium has addressed these issues satisfactorily, it is especially important that potential implementors of SGML understand them because they go right to the heart of the benefits of SGML.

The standards developers' first concern is a natural one. They have put a huge amount of effort, over many years, into creating a successful standard. Yet for many years SGML was largely ignored and sometimes derided by vendors, including some who have since joined the consortium. The authors of SGML are now wary of ceding to these vendors a leadership role in promulgating "their" standard. They also tend to be suspicious of any lobbying by vendors for changes to the standard itself.

Vendors, however, must be involved. Practically speaking, adopting SGML means using some vendor's tools. The more tools that vendors create, the more experience they will get in designing user interfaces that make SGML accessible, and the more users will reap the benefit. The biggest single hindrance to wide-scale adoption of SGML in the '80s was the paucity of SGML products that gave non-technical users access to SGML applications. Vendors themselves have now decided that SGML is a business to be in, and they are increasing their SGML product development activities. This should be welcome news.

Will SGML Open "Water down" SGML?

The other issue that concerns standards developers is that the consortium will compromise SGML by coming up with a "lowest common denominator" approach — something they can

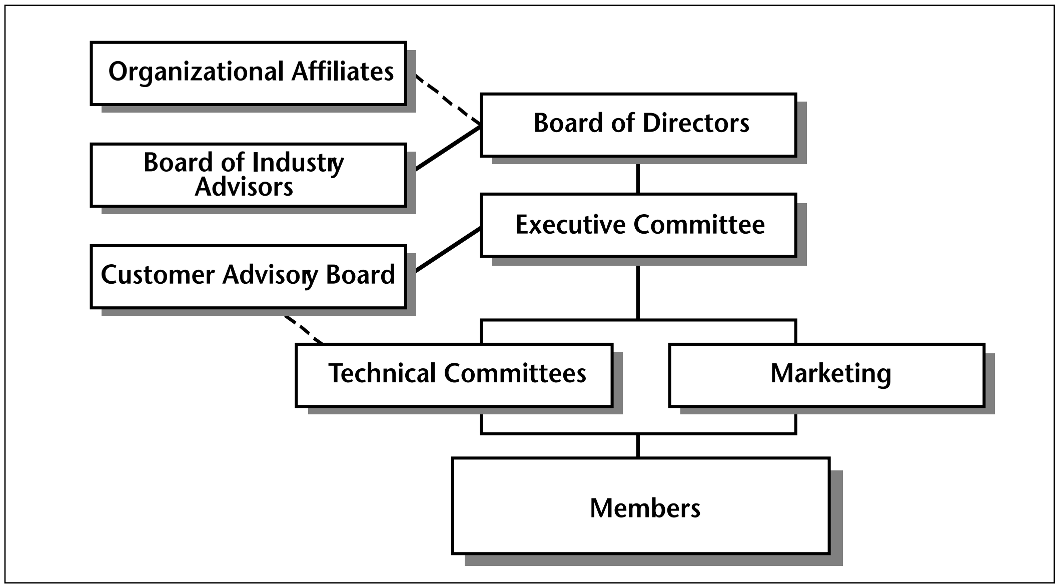


Figure 5 The SGML Open Organization

all support regardless of whether it solves users' needs or even conforms to the standard. The flexibility and power of SGML would be lost to users if vendors adopted such a strategy.

This, in our view, is a legitimate concern. None of the vendors *intend* to follow this path, but vendors and users alike share a natural tendency both to think about SGML in concrete terms and to fall back and reuse that which has worked before.

To apply SGML to real world problems, you must define the scope of each problem and apply (bind) a particular SGML solution to it. Once one finds a solution to a particular problem, it is natural to try to apply the same solution elsewhere, even to what may be very different problems.

Consider tables, for example. One can model tabular data in many different ways under SGML; some will be better suited for a particular application than others. Yet if one application of SGML (*e.g.*, CALS) deals satisfactorily with complex tabular material, users are apt to try to apply the same model to other applications, even if others would be more appropriate.[5]

The particular concern of the standard-makers is that vendors, too, may foist inappropriate SGML solutions on customers, especially if the vendors agree to support particular models and not others.

Should You Be Concerned About These Issues?

We think the consortium understands these issues and that they will work to avoid falling into such traps.

The SGML Open organization (see Figure 5) is designed to ensure a balanced program of activities. The Board of Industry Advisor's job is to help keep the consortium on track with its own charter as well as with other industry developments. Charles Goldfarb, a member of the board, the editor of the SGML standard, and an influential member of ANSI and ISO standards committees, will facilitate complementary relations between the standards bodies and the consortium. The Customer Advisory Board and the Organizational Affiliates (the GCA and the International SGML Users Group) will help keep user requirements at the forefront.

All in all, we are very encouraged by the wide support SGML Open has received already. We hope vendors invest more than words in advancing the consortium's agenda. Such a consortium cannot clear every obstacle that companies now face in adopting SGML, but as SGML implementations proliferate in the next few years, it can help vendors and users both to obtain the greatest payback with the least amount of unnecessary risk.

By promoting a standard vocabulary through which to describe SGML features and compare products, and by identifying issues that are best addressed by the SGML community as a whole, this consortium can foster stable growth and thus reduce risk for all companies that need open information.

How To Find Out More About SGML Open

For further information contact one of the acting officers, or call ***(416) 239-4801****.*

The current address for SGML Open is:

SGML Open
P.O. Box 7094
Boulder CO
80306

5. This is one of the reasons SGML is often confused with the DoD's CALS standards. SGML is one of the standards that the DoD has adopted as part of the CALS initiative. Their use of SGML is different, and not necessary applicable to other industrial uses. This is why industries (like the automotive) have their own SGML initiatives under way.

DOCUMENT QUERY LANGUAGES — *WHY IS IT SO HARD TO ASK A SIMPLE QUESTION?*

EXECUTIVE SUMMARY

Strategic Overview

- Today's databases were designed to handle fielded, mostly numerical data. Many features of these database systems cannot yet be used to manage libraries of text and mixed-media documents.
- Due to rapid growth of on-line text and document storage, many information-systems managers need to develop plans for managing databases of complex documents.
- This article addresses how trends in the development of document query languages can help answer this need.

Document Query Languages & Retrieval Systems

- Today's mostly relational databases are not optimized for text-processing, and they do not deal particularly well with text documents.
- There are several interesting reasons beyond text-retrieval why information-systems professionals should pay attention to trends in document management and retrieval.
- Documents employ deceivingly complex data structures that are difficult to model. Most computer programs deal with documents by abstracting only the data of most value to a particular application.
- Despite new tools that build queries automatically, formal query languages are still needed, for use both in interactive and embedded queries.
- Content-based queries start by assuming that the data within each document is unique. Full-text content searches are quite fast, but they are best used on relatively stable document libraries.
- Database-oriented queries are typically SQL applications managing DBMSs that contain or point to documents. They are particularly well-suited for revision management or workflow management systems.
- Structure-sensitive queries support precisely defined searches, and can be used to attach comments or formatting data to read-only documents. They require that the structure of the documents themselves be modelled with a language such as SGML.

Risks & Costs

- Few document-retrieval products require that source data be converted to proprietary formats. Most, however, create large indexes that must be stored on-line with the source documents.
- Some products, especially the structure-based ones, require that documents be converted to and stored in ISO-standard SGML format. More standards are needed to manage documents in this format; some that have been approved are not yet supported by commercial data-retrieval products.

Conclusions & Recommendations

- Modern document-retrieval technologies can make businesses more competitive in manyways, some of which are not widely acknowledged.

- Many such products can be introduced with minimal risk. Others incent companies to move faster in adopting data standards such as SGML. This introduces different risks, which managers must also monitor carefully.
- Different classes of queries benefit users in different ways. Their needs should be weighed carefully when designing document-retrieval systems or selecting applications with which to test them.

STRATEGIC OVERVIEW

Today, data is highly structured and carefully organized. Users can retrieve information quickly and easily. Applications share access to common databases, which each can manipulate through industry-standard application programming interfaces (API's). Right?

"Many of the more difficult—and arguably more important—databases remain to be built: those that manage information in document form."

Well, yes and no. So much progress has been made in database management in the past 20 years that it is easy to forget that software vendors tackled the easy problem first. Many of the more difficult—and arguably more important—databases remain to be built: those that manage information in *document* form.

In our previous issue, we observed that database systems and document systems have developed separately to date, with neither benefiting much from advances in the management of the other. This must change. Tomorrow's information systems will be hybrids, systems that must accommodate documents and their component elements with as much facility as they now manage data records.

How we get there from here is still a matter for debate. Progress is already being made on several fronts, however, and the question that remains is which of these efforts will prove most productive. Some are data-centered; they focus on how best to organize documents and libraries for retrieval:

- Developing and testing products based on new database paradigms (*e.g.*, object-oriented)
- Extending relational or other database structures to handle large or complex objects, such as documents, or new datatypes, such as animation and video
- Developing tools for linking into objects that reside outside databases
- Modelling documents themselves with standards such as SGML and HyTime
- Designing core utilities for managing multimedia documents

Others focus on breaking down barriers that separate users from existing data:

- Standardizing the syntax (or programming interface) of existing text query languages
- Hiding query functions generally behind more graphical user interfaces
- Adding support for new datatypes in existing query languages, such as SQL

We find such progress encouraging. In the meantime, however, information-systems managers face the issue of what to do about documents *today*. Products exist that enable users to query libraries of documents for information, but they differ greatly. Standards are evolving to support far more powerful queries against such libraries soon, but few of today's products support them.

Which of these standards will prove most useful in the long run? Which will mature in a timely enough way to enable your company to remain competitive in your ability to store and manage information? And which products, if any, should you adopt while waiting until these standards have been proven?

Document Query Languages & Retrieval Systems

What is a Query Language?

At its most basic, a query language is merely a formal way to pose questions. Most people manage even as children to master simple query languages. Consider Marco Polo, for instance, a popular version of the game "tag" often played in a swimming pool. With eyes closed, one player repeatedly asks the same question—where are the other players?—using a stylized query ("Marco"); the others answer just as formally ("Polo").[6] Another such example: Simon Says, where winning depends on alert responses to variations in command syntax.

"As we ask databases to do more for us, the languages we use to pose queries to them must change and grow with our needs. "

As they mature, of course, people learn to address more complex issues and to interpret and process more complex questions. Adults rarely use formal syntaxes in daily life, except in directing less intelligent agents such as computers and pets. Restricted vocabularies and formal syntaxes survive, however, in some highly structured professional environments, such as air traffic control and commodity trading.

In information processing, the best-known query language by far is *Structured Query Language (SQL)*, developed at IBM by E. F. Codd and C. J. Date and evolving over the years into ANSI and ISO standards. A nonprocedural language, SQL can be used in interactive queries or embedded into procedural database applications. Despite extensions, it remains essentially wedded to the relational database model which Codd and Date invented.

As we ask databases to do more for us, the languages we use to pose queries to them must change and grow with our needs. Today's mostly relational databases were never optimized for text-processing, so it should not be surprising that they do not deal particularly well with text documents.

A simple example: in the relational model, the *order* of database records is not particularly significant. In a text database, however, the sequence in which paragraphs, sections, etc., occur is of considerable importance, and tools are needed that can add, delete, merge, copy and split paragraphs, for example, without throwing the surrounding document into chaos.

Why Query Documents?

Until recently, many information-systems managers could afford to ignore documents. Before the 1980's, when the price of magnetic data storage plummeted, managing large volumes of documents on-line was relatively rare. Now even ordinary users have tens or hundreds of megabytes of disk storage at their desks, plus gigabytes more on shared file servers. In such circumstances, the chances of users losing documents—or not finding them when they're needed—increase greatly.

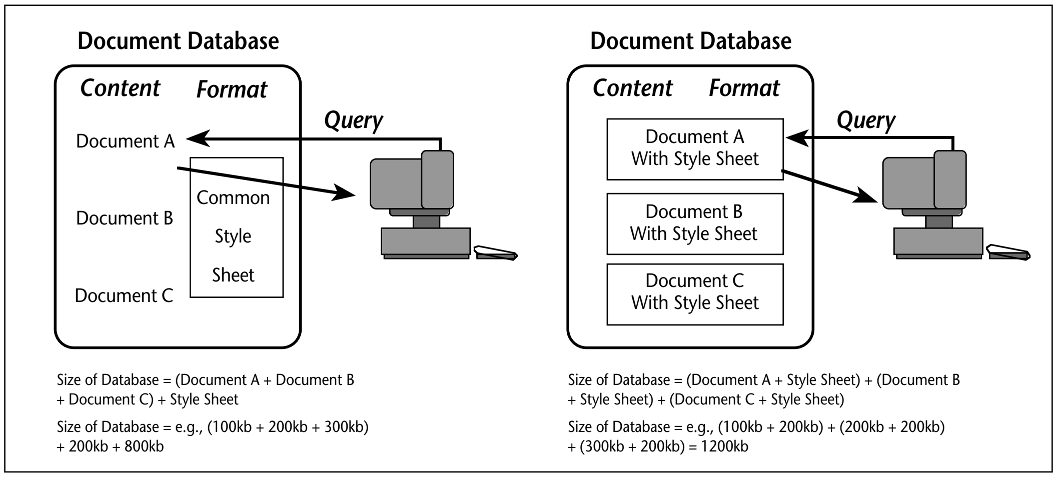
So locating data is the most obvious reason for querying document libraries. *Pull up all the correspondence on the Johannsen account.* Done. *Where's that article I saved on the Alzheimer's clinic in Detroit?* Found. *Was it Harriman that wrote me last March about the sprinkler system?* Confirmed.

A second trend complicates this problem: our documents themselves are growing more complex. Two years ago, people spoke mostly of *text retrieval*; today, however, we can seldom assume that documents consist solely of text. Documents may "contain" graphics or other datatypes (*e.q.*, equations) that cannot be represented clearly in simple ASCII. They may "contain" information such as editors' notes that are not rendered when the

6. The object of the game is to use these responses to locate and catch other players.

document is printed. And as the meaning of "document" stretches to include material that is published electronically, such documents may contain multimedia objects that are not even capable of being rendered on paper.

Especially as we cross this boundary, we discover reasons to query documents that are not limited either to text or to retrieval alone. Today we also query documents to update them, to index them and to annotate them. Suppose, for example, you receive a set of documents on read-only media, such as CD-ROM, and you want to allow employees to



Document Database
Content
Format
Query
Document A
Document B
Document C
Common Style Sheet
Document Database
Content
Format
Query
Document A With Style Sheet
Document B With Style Sheet
Document C With Style Sheet
Size of Database = (Document A + Document B + Document C) + Style Sheet
Size of Database = e.g., (100kb + 200kb + 300kb) + 200kb + 800kb
Size of Database = (Document A + Style Sheet) + (Document B + Style Sheet) + (Document C + Style Sheet)
Size of Database = e.g., (100kb + 200kb) + (200kb + 200kb) + (300kb + 200kb) = 1200kb

Figure 1

Re-using Linked Style Sheets

comment on them using electronic notes. How can you attach their comments to topics in the read-only document? One way is to program a "comments editor" to query, not for the text itself, but for a pointer to its location within the document. You can then use this location pointer to define a connection (*link*) between the comment and its subject, and build a document *browser* that displays both.

"If a universal query language can be developed for fielded data, what prevents us from doing the same for documents?"

Another intriguing (future) application is in document formatting. One reason people soak up so much disk space today is that every component of every document carries with it an armload of formatting instructions.[7] Suppose instead that there was a common formatting specification for all documents, and that we could "format" them primarily by building links between the formatting specification and document components. To build such links, we could issue queries to the document, such as *return pointers to all bulleted list items.*[8] These links could then be used by a desktop publishing program, for example, to render the document in PostScript on its way to a printer or typesetter.

What Makes Text Different?

If a universal query language can be developed for fielded data, what prevents us from doing the same for documents? The answer is not a simple one. One problem is that documents contain *natural* language, where there are many different ways of saying the same thing. This of course complicates the task of locating and retrieving relevant information.

Another problem is that the data structures occurring in documents are complex and difficult to model. We learn from early childhood to deal intuitively with books: to infer

7. On many modern publishing systems, the "WYSIWYG" document is two to ten times as large as the corresponding plain-text ASCII version.

8. This approach helps solve problems arising from the use of one document in two or more contexts. These not only include situations involving monitors with different aspect ratios or printers with different fonts, but also those where particular kinds of information in documents must be kept hidden from certain users. (Reasons for doing so run the gamut from guarding confidential data to accommodating user preferences.)

"When computer applications do model documents, however, they typically simplify the model, by abstracting only those structures most critical to the job at hand."

relationships between text and illustrations, to understand the role of emphasis and to accommodate the interruptions of footnotes, cross-references and page boundaries. Through experience, we navigate easily from table of contents to topic to index, or from cross-reference to photo to caption to referenced call-out, without giving much thought to how hard it might be to represent digitally the logical structures (links) that support such behavior.

When computer applications do model documents, however, they typically simplify the model, by abstracting only those structures most critical to the job at hand. In other words, we deal with documents by pretending they are less complex than they are, or, we might say, by disabling them in some way. *How* we simplify—that is, what features of documents we fail to reflect in our data model—sets our application apart from others that work with documents in different ways.

Examples of such compromises are easy to find:

- Many applications see documents as defined primarily by a single continuous strand (thread) of text, which can be represented by consecutive bytes of ASCII character data. In this model, two words are considered to near each other if stored in computer memory at nearby addresses; the last word in the body of Chapter 5 and the first in the title of Chapter 6 may be treated as consecutive, although they may fall on different pages and perform very different logical functions within the document.
- Some applications treat documents as primarily a sequence of self-contained units—that is, pages. In loose-leaf publishing, each printed page can be revised and re-issued as a stand-alone document; many document imaging applications also start from this premise. In such applications, a search for the phrase "top banana" might fail if a page break separates the two words.
- Others manage documents as ordered sets of individual lines of text. Many word-processing systems store documents in this fashion; some less sophisticated tools cannot deal gracefully with strings that span adjacent lines.
- Until recently, nearly all applications defined documents only in terms of what could be represented on paper. This excluded movies and sound as potential document elements. (The advent of multimedia applications has now forced many people to re-examine such restrictions.)
- Increasingly, document-processing applications treat documents as a hierarchy (or web) of functional elements. For example, a document contains chapters, a chapter contains sections, etc. Likewise, a paragraph may "contain" notes—even though the notes themselves may be relegated as endnotes to the back of a chapter, as footnotes to the bottom of a page, or as hidden comments displayed only upon command.

All these approaches mirror ways that we work with and think about different kinds of documents. We talk of the narrative thread within a novel; we copy pages from a

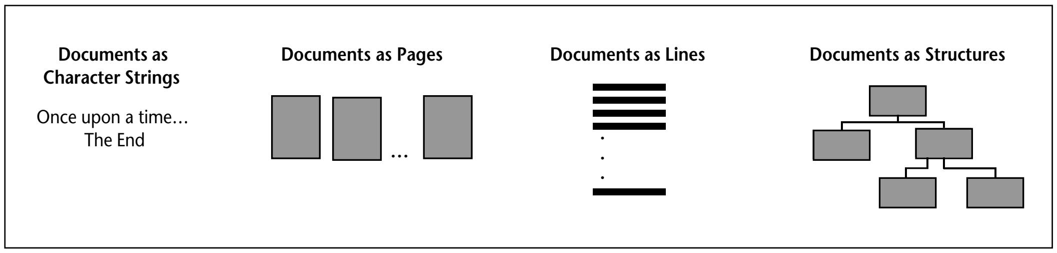


Figure 2 The Ways Applications Think of Documents

magazine; we press Delete Line on a word-processor; we revise an outline; or we arrange books on a shelf not by content but by size. We switch back and forth easily among these different ways of thinking about documents, much more naturally than computers can.

Where document-processing and -querying products usually fail, in fact, is in mixing models within the same application. Most 7-year-olds could, if asked, identify "the first person named on the third page of Chapter 12," but processing the same query by computer requires *(a)* that we understand the logical structure of the document (chapters); *(b)* that we track capacity constraints on the processing of the document by a formatting system (pages); and *(c)* that we not only isolate proper nouns (names), but also determine which ones refer to people, not places. Today's database systems, and the query languages designed to work with them, are not capable of any of this.[9]

Who Builds Queries?

"Among the factors shaping the design of query languages are our assumptions about who will use the language—i.e., who will build the queries."

Among the factors shaping the design of query languages are our assumptions about who will use the language—*i.e.*, who will build the queries. From these assumptions and others, we predict what forms of queries users will need, and we build systems optimized to handle them.

For database analysts, therefore, we are likely to develop extensions to query languages they already know and use. But for multimedia developers, whose "documents" may contain no text at all, we might instead start fresh, building a system that calls as-yet-undefined utilities for processing as-yet-unknown multimedia data objects.

In both cases, we would probably favor a English-like syntax that such analysts or developers could learn and use easily. But as we automate more tasks, we also see more computer applications generating queries programmatically. When designing query syntaxes with such APIs in mind, we typically place less emphasis on making queries readable or "intuitive" to human users.

Other assumptions, too, color the design of query systems. As query systems begin to take document structure into account, it is becoming increasingly common for document-retrieval systems to assume that all documents are compatible with, or convertible to SGML. Although we believe strongly in the future of SGML, few SGML libraries yet exist, and tying such systems to it now may delay their acceptance.

Are Formal Syntaxes Still Needed?

None of this means that formal query languages are always necessary. Some users need nothing more than the ability to select among predefined queries; for others, a graphical user interface that prompts for query terms may be sufficient. Natural-language and query-by-example techniques may serve some users better than direct access to a formal query language. [10]

But there will always be exceptions. Over time, some users respond to query systems by asking new and better questions. Casual users grow to become power users. Vendors know this, and therefore typically implement such casual-user interfaces as applications atop a more robust query-processing system, one that also allows the framing of complex queries in a formal syntax.

9. Consider the consequences if the query processor also had to handle modifiers such as "in the book with the unicorn logo on each page," "in the draft on Alan's hard drive," or "after the reference to a popular mountain resort."

10. Few natural-language query systems perform any semantic analysis of queries. A queries such as "Tell me about baseball in Japan" often return all documents that mention either baseball or Japan. Modern relevance-ranking algorithms, however, often compensate for this by sorting results by number of query terms found, number of "hits" on each term, occurrence of terms in the query phrase order, etc.

One should not, therefore, shop for document-retrieval systems only on the basis of their casual-user features. It is, however, appropriate to ask questions such as:

1. What kinds of queries must users be able to perform?

2. What kinds of query languages must applications be able to support to perform such functions for them?

3. To what degree can users' query needs be met by such applications?

4. To what extent do users themselves need to use query languages directly to perform the queries that the applications do not support?

Contending Technologies

Note that we speak here of languages in the plural. We do so reluctantly, for ideally no firm would need more than one.

In the real world, however, people envision and describe documents in many different ways. These include references phrased in terms of:

1. Artifacts of the document in its formatted state (page number, etc.),

2. Artifacts of the document in its pre-formatted state (filenames, etc.),

3. Its logical structure (chapters, etc.),

4. Its data content (characters, etc.), and

5. Externally supplied attributes (version, subject).

In theory, document query languages can make use of any or all of these "handles." In practice, however, even hybrid retrieval methods seem to favor one set of criteria over others. So great are these differences that document-retrieval products themselves and

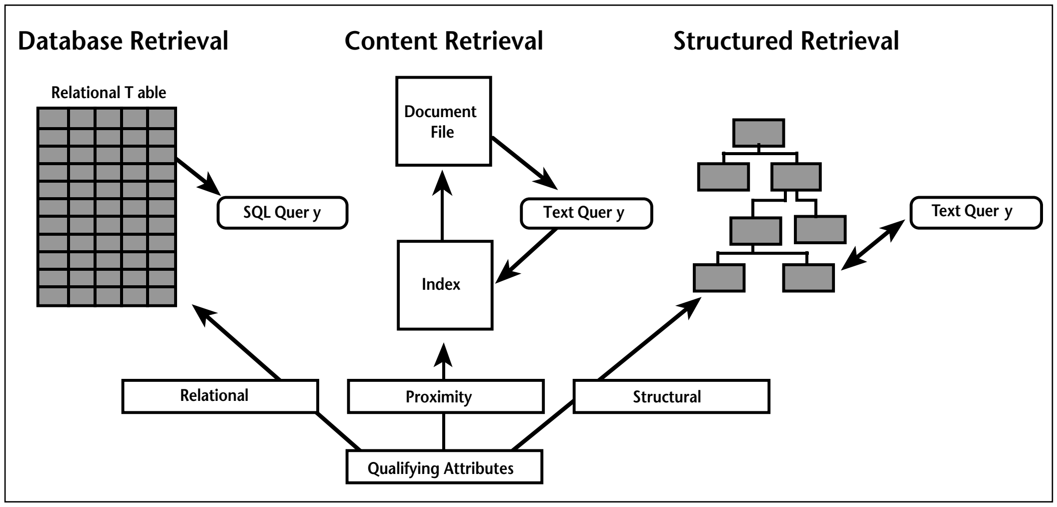


Figure 3

The Three Types of Searching

the query languages they employ are also quite different. We can therefore categorize such products by how they have us describe the documents we want to retrieve.

Because our interest is primarily in electronic documents, we will not address here products that deal primarily with the formatted representations of documents (*i.e.*, paper pages, microfiche, etc.). Nor will we bother to address techniques for organizing and retrieving documents by filename and directory placement, since for most people, this represents the *status quo*.

Instead we will compare approaches based on content, structure and externally defined attributes—the three "handles" most used in document retrieval today. (See Figure 3)

Content-based Query Languages

A particularly popular technique now is content-based retrieval. Content-based approaches start with an assumption that what most distinguishes one document from another is the data inside. Primarily text-oriented, content-based methods today range from simple line- or record-based pattern matching to more complex algorithms for analyzing document semantics.

"A problem that one must often overcome when loading documents into full-text repositories is hiding markup and formatting artifacts from the indexing engine. "

The most straightforward of these opens each file or record and matches against one or more user-specified text strings or patterns. The UNIX *grep* tool, for example, takes as input a *regular expression*, whose syntax is well-known in the UNIX community. On DOS machines, Norton Desktop's SuperFind and similar utilities follow the same approach. Searching large libraries of documents by this method is inefficient, but as the cost of computers and storage comes down—and more people find themselves with devices that scan megabytes of text in just a few seconds—such utilities are proving popular, especially for managing personal libraries.

Such techniques break down when applied to larger libraries or slower storage devices. (CD-ROM raises both concerns.) To speed up searches, most content-based applications first build an index to every significant word or topic in the library; *full-text* queries work by matching against this indexed list of all occurrences, rather than opening and reading every document.

Full-text search engines retrieve data very quickly. But building the inverted index that such a process requires may take hours or even days.[11] *Incremental indexing* (in which only new or changed data is re-indexed) helps minimize redundant processing; also, the indexing step can run at off-peak times, perhaps overnight. Still, the requirement to build an index means that full-text engines are usually best reserved for libraries that are relatively static.

Full-text engines may perform little semantic analysis of documents, other than deciding what to treat as words. *Stop lists*, however, often winnow out words that are too common to be useful in search expressions. (Conversely, some applications use *go lists*, which specify which words *should* be indexed.) Some full-text vendors now offer thesaurus capabilities; these enable users to search for topics instead of mere words. Some products offer more sophisticated ways to define such topics, allowing users to assign likeliness-of-relevance attributes to different terms or phrases.

Many products use different *relevance-ranking* techniques to sort query results. *Weighted Boolean* techniques take into account how often the query terms appear in the document, and how rarely they occur in the library as a whole. *Fuzzy Boolean* searches for "Simon and Garfunkel" would return articles about either singer, but would list as more relevant those that mentioned both.

New *query-by-example* features allow users who find a document or passage that interests them to ask, in essence, "show me more documents like this one."

A problem that one must often overcome when loading documents into full-text repositories is hiding markup and formatting artifacts from the indexing engine. Many cannot deal with hyphenated words, or with acronyms such as "AT&T" that contain non-alphanumeric characters; others choke on accent marks. Words split between two lines by a word-processor's justification algorithm may also be handled incorrectly in the indexing

11. Modern full-text engines can process as much as 800 megabytes of data per hour. But they remain extremely sensitive to differences in operating systems performance. So procedures that take minutes on one system may run for several hours on another.

process. Documents containing typographic or structural markup, such as troff- or SGML-tags, must be processed before indexing to prevent these tags from being treated as content.

Database-oriented Query Languages

Sometimes we can distinguish between documents by their content; sometimes we cannot. Consider libraries that hold two or more versions of the same document. Different drafts or revisions are often hard to tell apart, especially on computer screens where both cannot clearly be viewed side-by-side. Likewise, maintenance manuals for similar pieces of equipment often look alike; users are apt to retrieve the wrong one if all they can do is match on phrases such as "wheel assembly."

"Powerful though they are, full-text search engines are often even less sensitive to the structure of documents."

The systems we create to handle such situations use techniques familiar to information-systems professionals. Database-oriented query languages rely upon tables of document attributes (metadata) to describe and locate documents, and most are implemented as relational database applications. This approach works best where it is important to ensure that users receive the *right* document or version thereof—unlike systems designed to help users browse for *any* document whose content might be of interest.

Documents, however, tend to be big—and until recently, relational databases offered poor support for such so-called *binary large objects (BLOBs)*. Vendors still have not solved all the problems that arise from putting documents *inside* relational DBMSs; in fact, they have only recently tackled the most basic one, that of making records large enough to hold documents at all.

For the most part, these systems assume that users will locate documents by their attributes, not by searching for characters or words inside the documents themselves. (On many RDBMSs this is still quite slow.) It is critical, therefore, that such attributes be assigned correctly when documents enter the database. Some document management and imaging systems automatically extract some such data from the documents themselves. Most prompt users to enter or verify such data whenever new documents enter the library.

Systems of this type often accept SQL directly, sometimes hiding it behind a graphical user interface that helps nontechnical users build queries automatically.

Database-oriented products are often sold as revision management systems, or they are tightly integrated into workflow management tools. Features to look for in such systems include those that minimize redundancy in the storage of different versions of the same document, as well as support for managing documents as a structured hierarchy of components, where each chapter, illustration, etc., is stored and managed as an individual file or record.

Structure-based Query Languages

As a rule, when database-oriented products allow documents to be defined as a structured hierarchy of components, that structure does not run very deep. Chapters, illustrations or tables may be handled as separate objects, but not paragraphs, captions, or table cells. One can query perhaps for all the chapters that changed since a particular date, but not just the changed hazardous-material warnings or the revised cross-references.

Powerful though they are, full-text search engines are often even less sensitive to the structure of documents. Sometimes this yields unexpected results. Searching for a document by title might also return other documents that mention that document by name, such as in a bibliography or footnote. Or searches for articles *by* writer "John McPhee" might also return articles *about* him.

Several text-retrieval products handle *some* problems of this type by combining content- and database-oriented query techniques. To do so, however, we must program the indexing engine to recognize and extract certain fields (*e.g.*, author) from source documents—which means we must be able to predict the kinds of questions users will ask. Often we cannot.

In some cases, too, we want to search on information that we cannot justify extracting from every document during indexing. Today, for example, we may decide to compile a glossary by finding all the terms defined within a document. Tomorrow's task may be to extract and sort entries from several different glossaries into a master version. An efficient way to proofread cross-references may be to extract each one and list them all in checklist form. A particular user may want to see only those news articles in which a certain word or topic appears in the first four paragraphs.

Search engines that address such queries need to be able to distinguish among the form or function of different parts of a document; that is, they must be sensitive to the document's *structure*. Designing systems around this requirement means the structure must be modelled in some fashion.

"When we read, we draw meaning not only from the language used to express concepts, but also from the way such concepts are organized and presented to the reader. "

This is most easily implemented when all documents in a library are structured in the same way. But this is rare. As a result, most work in this field presumes that the system must be able to handle any document and any document structure.

Enter SGML, with its mechanisms for both (a) defining structures and (b) validating conformance to them. To define the structure required for any class of documents, one can write an SGML *document type definition (DTD)*. This DTD, aided by an *SGML declaration file*, also specifies how that structure is reflected through the *markup* used to tag individual documents (*instances)* within this class.

With tools such as these, one can build systems that reflect a richer understanding of documents and the roles they play. When we read, we draw meaning not only from the language used to express concepts, but also from the way such concepts are organized and presented to the reader. Retrieval tools that are sensitive to such organization add to our ability to understand and use the information we retrieve from documents.

Such tools also enable us greater freedom to enrich documents with author- or reader-defined attributes. We can specify not only who wrote a procedural manual, but also who signed off on a particular task description. We can provide several different captions for the same drawing, and specify through attributes when to use each one. We can not only tie a particular file to a style sheet, but can specify precisely which paragraphs or list items are to be formatted in unique ways.

Indeed, it is this approach that the ISO community has taken for defining how documents should be formatted or referenced. In May ISO takes a vote on a draft standard, DSSSL,[12] that describes how to specify formatting of SGML information, by linking formatting instructions into an undisturbed SGML text file. Last August ISO approved a related standard, HyTime,[13] that established protocols for linking hypertext and multimedia objects, including SGML-tagged text.

12. DIS 10179, Document Style Semantics and Specification Language (DSSSL).

13. ISO/IEC 10744, Hypermedia/Time-based Structuring Language (HyTime).

Both the DSSSL and HyTime standards include specifications for query languages; they differ because of differing assumptions as to the needs of the "users" of each language. (In both cases, these users are most apt to be other computer applications.) DSSSL is designed to handle structured information, for example, an ordered hierarchy of document components where queries return sets of objects in document order—that is, in the order in which they would occur in print. HyTime, on the other hand, takes in multimedia objects, where order is temporal as well as spatial; it does not presume, for example, that a "document" is printable. Instead of returning sets, HyTime's query language, HyQ, returns ordered *node lists*, which (unlike sets) permit duplicate entries.

Both the DSSSL Query Language and HyQ are remarkable in their ability to represent complex queries, such as "locate numbered footnotes only in sections that contain fault-maintenance procedures developed at the Denver plant." Neither syntax, however, has yet been implemented and tested, and neither can be evaluated against other search technologies until commercial products that support them become available.

Also years away are extensions to SQL to handle data found in list and node structures, not just in tables. In the meantime, the aviation industry in particular has begun promoting what it calls Structured Full-text Query Language (SFQL). SFQL extends SQL with such features as proximity search, fuzzy matching, relevance ranking and extended data types, including SGML text, TIFF images and CGM graphics. Prototypes have been built by both General Electric and Aerospatiale, and servers are now commercially available.

Strengths and Weaknesses of Different Approaches

Each of these three types of query methods have unique costs and prerequisites. Querying by content typically requires that one first build a full-text index. Querying by document attributes requires that such attributes be loaded and validated, often manually. Querying by structure relies on users specifying such structure, for example with SGML tags.

The Future of Text Retrieval?

Content-based text retrieval may be about to take a leap ahead with the advent of retrieval engines rooted in artificial intelligence and vector algebra.

A very real cost of many of today's content-based products is the time needed to build application-specific thesauruses to support topic searches. A less tangible (but very real) problem is the difficulty users face in framing effective queries as standard Boolean expressions. Especially hard to handle are requests such as *find documents about ham not eggs, but don't exclude any just because eggs are mentioned*.

Several companies are engaged in research to address these problems. Particularly promising are reports by HNC Inc., a San Diego start-up, of progress in training computers to classify documents, words and phases by subject matter. Its *MatchPlus* technology, funded in part by the Defense Advanced Research Projects Agency (DARPA, now just ARPA), uses neural-network learning algorithms to quantify related terms by how frequently they occur in the same document or phrase.

MatchPlus represents topics with what it calls a *context vector*, a fixed-length vector in multi-dimensional space. Each of the approximately 300 dimensions can represent different properties, or *features*, that may relate to the subject. In a library of news articles, for example, sample features might include *fragrant, hot, flower, male,* and *blue*.

"Structure-sensitive queries allow us to ask the most precise questions about how such words or topics might be used."

The benefits are similarly varied. Query-by-content methods cast the widest net, using inverted indexes to quickly locate any document that mentions a particular word, phrase or topic. Structure-sensitive queries allow us to ask the most precise questions about how such words or topics might be used. Queries on document metadata (attributes) are fast *and* precise; with them, we are more likely to arrive at the particular document or version best suited to the task at hand.

In light of these differences, vendors find different ways to combine approaches. Some, like Fulcrum Technologies, have come up with ways to extend SQL to include full-text queries. Berger Levrault/AIS decomposes SGML documents into relational database records.

Today's Document-Retrieval Products

In fact, many of today's commercial document-retrieval products are hybrids of some sort, though most are clearly rooted in one of these three approaches to document management.

We can trace the management of large libraries of documents with relational database technology to the development of imaging systems by vendors such as FileNet, IBM, Kodak and Wang. Systems designed for electronic documents followed by a few years. Some, like the one developed by FrameMaker reseller Workgroup Solutions, were designed as general-purpose document-management systems that can track files of any type. Others, like Westinghouse's PATHWAYS Interactive Publishing, are far more task-specific or require documents to be in SGML format.

Boss Logic, Documentum, Interleaf, Micro Dynamics and Xyvision also offer products that support queries of this type. Many are evolving into workflow management products, like InfoDesign's WorkSmart software, which was originally developed for the U.S. military under the Joint CALS program.

Full-text retrieval products have a long history, growing out of research funded by

Once the computer is trained to recognize features, it can analyze large libraries of documents and assign such vectors to every word. Words with different meanings would get very different vectors; however, similar vectors would go to words that are similar in meaning (*car* and *automobile*, for example) or in context (*car, passenger, swerve* and *stoplight*).

Vector algebra can then be used to produce aggregate vectors for entire phrases, paragraphs or documents (*see figure*). To retrieve documents, the same process is used to assign a vector to each query. Queries need not conform to a particular syntax; any text string—even whole documents or selected paragraphs within them—can be used to define a query. The retrieval process takes the query and retrieves documents with similar vectors.

This method works especially well on *ham-but-not-eggs* queries. But what makes this technology especially exciting is the progress HNC reported recently in classifying documents *without* first assigning properties to each dimension. Since training computers to read words the way we do is time-consuming and costly, HNC has begun allowing the computer itself to decide which features each vector should represent. Although human observers might be hard-pressed to explain many of its choices, they proved surprisingly effective in classifying words and retrieving documents, without the up-front costs.

HNC has not yet announced plans to license this technology, but it is already building prototype applications for both commercial and government use.

defense and intelligence services. Market leaders Information Dimensions (BasisPLUS) and IBM (STAIRS) have largely set the standards here. Fulcrum Technology's software is now found in a number of electronic publishing products, including Interleaf's WorldView; Verity and Personal Library Software (PLS) are also established vendors whose search engines are often bundled into other vendors' products.

"Another new product certain to boost awareness of full-text search is AppleSearch, a new groupware product from Apple's Enterprise System Division. "

In this category, however, the spotlight today is on new players. Adobe, a powerful force in publishing thanks to its PostScript technology, enters the field soon when it begins shipping Acrobat, which by year's end should offer a Verity search option. Unlike other document *browsers* on today's market, which re-format documents to fit the computer screen or window, Acrobat will use a stripped-down version of PostScript to retain the appearance of the printed page. Adobe recently nodded to the importance of document structure in announcing that Acrobat's yet-to-be-released Portable Document Format will be extended within a year to reflect the SGML structure of documents.

Another high-profile newcomer is WAIS, Inc., a company newly formed to carry on the development and support of Wide Area Information Servers (WAIS). Launched two years ago by a consortium of Thinking Machines, Apple Computer, Dow Jones and Peat, Marwick, the first WAIS was a public-domain "freeware" implementation of draft ANSI standard Z39.50. Already been installed on over 350 servers worldwide, WAIS drives many of the document servers accessible over the Internet.

A key WAIS feature is *relevance feedback*: users can select any document returned by WAIS and ask for others like it. WAIS's search capabilities have otherwise been primitive to date, but the planned commercial version will remedy this.

Another new product certain to boost awareness of full-text search is AppleSearch, a new groupware product from Apple's Enterprise System Division. In true Macintosh fashion, users mark documents for indexing by dragging them into a special "folder," which represents a directory on an AppleShare server. Apple's XTND file-converters extract text from documents for indexing and viewing; users see only text, since there are no links from the index back to the source application. *Reporters* (query scripts stored as a named icon) can be created and assigned to retrieve data on a regular basis; users can also copy their reporters and share them with others in their workgroup.

THE "RELIGIOUS" WAR OVER QUERIES

The most contentious debate in document retrieval today boils down to one issue: should query languages be optimized to fit the user or the data?

On the one hand are the proponents of languages such as HyQ, defined in an appendix to the 1992 HyTime standard. Backers of this approach argue that documents today — especially multimedia documents — are too complex to be modelled under anything other than an object-oriented paradigm. Queries, they say, must be designed to navigate through a web of linked objects, even traversing into and returning results from document components stored in notations yet unknown.

In the opposite camp are those who hold that document query languages will only achieve wide-spread use if they are user-friendly, not data-friendly. The market today for SQL tools and applications dwarfs other markets, even though there is far more information stored in documents than in fielded databases. This group argues that users are better served by building upon what

In the absence of standards other than SGML, only a few commercial products have yet emerged to handle structure-based queries. A notable exception is the Pat full-text database system from Open Text Corporation. Originally developed in support of the on-line version of the Oxford English Dictionary, Pat indexes SGML documents and provides an applications programming interface with which to query both their content and structure. The products of Berger Levrault/AIS are also based on Pat technology.

DynaText, from Electronic Book Technologies, is a document browser that accepts native SGML documents. Using a unique query algorithm, DynaText first samples documents for terms named in the query, then applies the query itself against this subset library. DynaText also allows application developers to provide "prepackaged" queries for any class of documents (such as "find all documents where author is X and <warning> contains Y").

Alliance Technologies also offers a suite of tools, under the name TextManager, for managing libraries of structured documents. Including are tools for identifying structural elements within documents not already tagged in SGML.

RISKS & COSTS

There are many traps associated with document query and retrieval solutions. Perhaps the easiest to fall into is underestimating the impact they can have on how you do business.

This danger shows up in many ways. One is resistance by users. In our experience, managers often place a high value on expanding access to information, but their employees do not. Many, in fact, feel threatened by document-retrieval technologies. Such technologies empower users, but in doing so they disrupt informal political systems, in which power often rests with experienced employees who know best where information resides. One of the biggest risks in implementing document retrieval systems comes from ignoring such politics and inviting revolt, often led by the firm's most influential employees.

Other companies err, however, by pigeonholing these technologies in applications so narrow that few employees benefit. Firms that invest in document-retrieval technologies

they already know, even if it means accepting some (temporary?) limitations in the kinds of searches they can perform against such data.

SFQL, which layers text-oriented functions upon SQL, has become a lightning rod for this debate. This is unfortunate. Despite its ambitious title, SFQL was designed for a particular text-retrieval application, and as such it is saddled with limitations (such as handling only ASCII text) that unfairly cloud the larger issue.

The "religious" debate over these issues may be entertaining, but in our view it has gone on too long. Languages such as HyQ have an important role to play in managing complex databases. We do not, however, accept the argument that SQL can never be effectively extended to deal with such data. It is true that SQL arose from the relational model, but the physical structure of a document database need not mimic the logical view that SQL presents to users. We see no reason why an extended SQL cannot coexist with HyQ: one as a general-purpose "front-end" for retrieving useful information from all forms of databases, the other a back-end programming interface for managing the complexities of each underlying document-based repository.

should look for applications that scale up well, so that the costs of getting into this game can be offset by distributing the benefits more widely.

In our previous issue, we discussed in general terms another risk, that of protecting one's investment in document management technologies through open information and industry standards.

We will not repeat that discussion here, but the same issues apply. Few document-retrieval vendors still require that you convert source data to proprietary formats that may soon be obsolete and unsupported. Some, however, require documents be converted to SGML. While most companies, in our view, should have a strategy for migrating to SGML over time, doing so without careful planning is almost as dangerous and costly.

The good news is that vendors are responding, if slowly, to customer demands that they be allowed to read, create and verify SGML documents. What we still lack, however, is a full set of standards for building, accepting, editing, managing, converting and formatting them. Standards such as DSSSL must not only be adopted, like HyTime, but must yet be understood by publishing professionals and supported by a range of publishing products. Only then will many firms feel safe entrusting libraries of "mission-critical" information to SGML.

This bears on document retrieval in another way, because while some search technologies require SGML, others still discourage it. Especially when evaluating content-based products, one should look not only at *whether* they support SGML but how; be especially wary of products that implement it in shallow, ham-handed ways that trivialize the importance of document structure. Structure-sensitive query technologies, on the other hand, often require SGML—perhaps before you are ready.

CONCLUSIONS & RECOMMENDATIONS

Modern document-retrieval technologies can improve businesses in many ways. Some are pretty obvious. General employee productivity among information workers improves with faster access to information. Customer service and customer satisfaction improve when employees who deal with customers can quickly pull together all the documents and data they need to address a customer's concerns.

Other benefits are more subtle. Redundant work is reduced through more effective reuse of existing documents and information. The quality of employees' work is enhanced through easier and wider exposure to examples of good work. Training effectiveness rises when the recently trained can review in an instant the materials and documents they covered in class.

Still largely unrealized are other benefits that *should* flow from new document-retrieval technologies. The cost of storing documents drops as formatting specifications are removed to a separate file, there to be shared among other documents. Hypertext webs develop as employees link documents to others that help to explain them. Employee writing styles become crisper; rather than repeat or rephrase information in other documents, they merely link the new document to them.

Today's document-retrieval technologies have evolved to the point where they offer real value across a wide range of applications. Most can be implemented with little impact on authoring practices (although some products, especially those designed specifically to direct employees to task-specific information, require more of authors than others).

Some retrieval products take only SGML-tagged documents as input. In some cases, such requirements will incent companies to adopt SGML more quickly; in others, they will delay implementation of a document-retrieval strategy, until the editing and formatting of SGML documents are better supported by standards and by commercial products.

Before selecting and implementing a document-retrieval strategy, you should:

1. Identify and analyze existing business practices that could benefit from faster information retrieval.
2. Verify that the information required to support these practices is available in document form. If it is not, determine how such a library could be created and maintained.
3. Determine which employees or departments can make the best use of such information. Consider such factors as their role in the flow of information, their experience with on-line documents and their attitudes toward technological change.
4. Consider whether the benefits you seek would be more likely to derive from faster browsing of existing data, more directed access to appropriate versions of documents, or an ability to specify more precisely the kinds of documents needed. Focus on content-, database- or structure-oriented technologies accordingly.
5. Examine products that employ such technologies and explore whether any address your needs. Pay special attention to assumptions these products make about the condition or form of your documents, the frequency with which libraries must be updated, and any ongoing expenses you must incur in managing such libraries.
6. If possible, identify applications that are small enough to serve well as testbeds yet large enough to justify the conversion and tool costs required for pilot programs.
7. Perform a cost-benefit analysis that includes both short- and life-cycle costs, as well as spreading out costs over time. Validate by preparing similar plans for other systems and under other assumptions. Consider different potential pilot programs as part of this analysis.
8. Implement in a manner that tests your assumptions about the technology itself and how introducing it can affect your business practices.
9. Scale up the project over time, modifying or fine-tuning it based on your experiences in the pilot phase.

Chip Canty

Corrected Illustration From The Last Issue

Due to a translation error the '**TI**' and '**T2**' were left out of Figure 4 in the article on imaging, document & information management systems. The complete illustration is reproduced below.

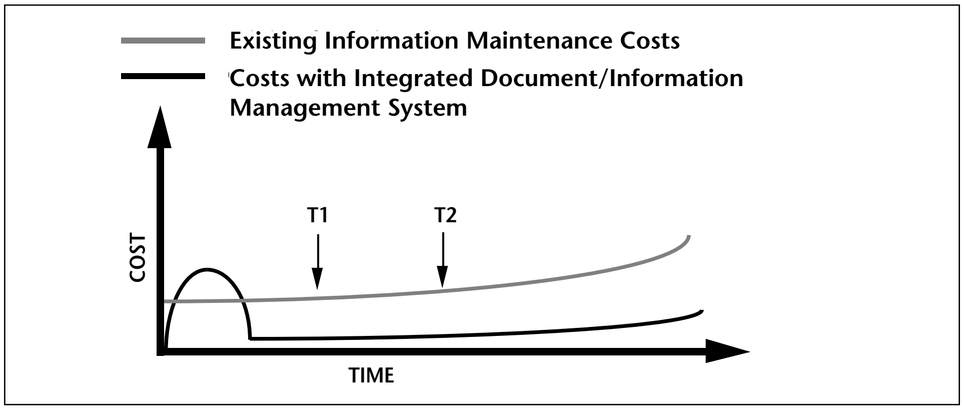


FIGURE 4

Relative Initial and Life-cycle Costs

GCA & PTM ANNOUNCE NEW 'DOCUMATION' CONFERENCE.

The Graphic Communication Association (GCA) and Publishing Technology Management (PTM) have announced a major new conference and exposition to focus on document-related technology and the convergence of document, data, and image management. The conference will be chaired by Frank Gilbane (your editor). Documation '94 will be held at the *Westin Century Plaza in Los Angeles, February 21-25, 1994.*

The focus of the conference is on the application of document technology in corporate information management strategies. This includes all aspects of document creation, document management, and document distribution, as well as the role of documents as interfaces to computer applications and databases, and as an architecture for managing multimedia information. The conference will be complemented by a formal exposition that will serve as a major event for vendors to demonstrate products and services that relate to document automation.

The Mission

The goal of Documation '94 is to provide managers the information they require to develop strategies and approaches for the next decade to address the increasing need to integrate document systems with enterprise information bases. The conference will draw upon industry experts, leading vendors, and successful users, providing a comprehensive environment in which attendees can gather all the information they need to begin to make strategic decisions.

The Program

An Industry Advisory Board, made up of twenty four leading companies in document and information system solutions, is currently reviewing program content and format. The conference will cover four areas:

- Imaging, document and data management — how these technologies relate and compete, what core technologies and standards are involved, what the business challenges and risks are, and how interoperability can be accomplished.
- New document system technology — multimedia, compound document architectures, client server architectures, middleware, and operating systems.
- On-demand document information — electronic distribution, demand printing, and mobile information access.
- Vertical industry document system applications — issues and case studies in telecommunications, pharmaceuticals, aerospace, insurance, and publishing.

Further Information

To receive more information on the conference, or to receive an exhibitors kit, call Marion Elledge at (703) 519-8160. If you have proposals for topics or speakers, call Frank Gilbane at (617) 643-8855, or fax your proposal to (617) 648-0678. We will keep you up-to-date as the program develops.

CALENDAR OF EVENTS

Below is a selection of key events covering open information and document system issues. There are many other conferences and shows covering related topics. We will attempt to keep this list to those events that focus on areas most directly related to the areas covered in our report.

Seybold Paris. June 1-3, 1993, Paris, France. A new Seybold conference and exhibition on electronic publishing. Call +44 32 341 0561, Fax +44 32 341 0279.

Information & Technology Week. August 30-September 3, 1993, Anaheim, CA. GCA tutorials and seminar. Call (703) 519-8160, Fax (703) 548-2867.

CALS Europe '93. September 22-24, 1993, Berlin, Germany. Conference and exhibition on CALS-related activity in Europe. Call (202) 775-9556, Fax (202) 775-8122.

CALS Pacific '93. October 1993. Conference and exhibition for CALS activities in the Pacific Rim. Call (202) 775-9556, Fax (202) 775-8122.

Seybold San Francisco. October 20-23, 1993. San Francisco, CA. The enormous computer publishing exhibition and conference. Call (310) 457-8500, Fax (310) 457-8510.

CD-ROM Expo. October 27-29, 1993, Boston MA. Conference, tutorials, and exhibition. Call (617) 361-8000, Fax (617) 361-3389.

CALS Expo '93. November 1-4, 1993, Atlanta, GA. The annual conference and exhibition. Call (202) 775-1440, Fax (202) 775-1309.

Hypertext '93. November 14-18, 1993, Seattle, WA. Conference covering research in applications of hypertext-related technology. Call (212) 869-7440, Fax (212) 944-1318.

Explor. November 14-19, 1993, Denver, CO. The annual conference and exhibition on electronic printing systems. Call (310) 373-3633, Fax (310) 375-4240.

TOPICS TO BE COVERED IN FUTURE ISSUES

The subjects listed below are some of the areas we will be covering, in no particular order. If you have an opinion about which topics you would like to see added or covered sooner rather than later, let us know.

Electronic Distribution — Does One Size Fit All? Who Are The Players? What Are The Options? Are Pages Important?

Documents As Interfaces — Is This An Option For Today? What Will The Future Bring?

SGML & Presentation Interchange — What Standards Are Available Or Appropriate? (DSSSL, OS/FOSI, HyTime, ODA, etc.)

Document Management & Databases — How Do Current Document Management Products Integrate Database Technology? How Will Future Products Do It?

Authoring Systems — Do You Need Different Kinds For Different Media?

"Middleware" — What Is This Layer Of Software In Between Operating Systems And Applications? Is It The New Proprietary Trap? What Does It Mean To Your Decisions About Document Systems?

ISO 9000 — What Kind Of Document Management System Do You Need To Meet This Quality System Standard?

Open Systems & Client Servers — What Are They? How Do They Relate To Document System Technology?

Document Elements & Distributed Objects — How Do They Relate To Each Other?

CALS & IETMS — What Are They? How Do They Influence Open System Technology?

Imaging Technology — How Is It Evolving?

The Airframe And Airline Industry's Strategy For Sharing Product Information — What Can You Learn From It?

New Drug Applications — What Document System Strategies Make Sense For The Pharmaceutical Industry?

Object & Relational Databases — Which Approach Is More Suited To Your Document Systems Needs?

Order Form

o Please start my subscription to: The Gilbane Report on Open Information & Document Systems (6 issues). Back issues available for $45.

U.S.A.: $225 Canada: $232 Foreign: $242

Please send me additional information on: o Consulting Services o Special Reports o On-site CALS Strategic Planning Seminar o On-site Open Document System Strategic Planning Seminars

o My check for $ __________ is enclosed o Please bill me

o Please charge my credit card o MasterCard o Visa o American Express

Name as it appears on card __________________________________ Number ________________

Signature ____ Expiration date____________

Checks from Canada and elsewhere outside the U.S. should be made payable in U.S. dollars. Funds may be transferred directly to our bank: Baybank Boston NA, 175 Federal Street, Boston MA 02110, S.W. code BAYBUS33, into the account of Publishing Technology Management, Inc., number 1444-89-63. Please be sure to identify the name of the subscriber and the nature of the order if funds are transferred bank-to-bank.

Name ____ Title__________________________________

Company ______________________________________ Department___________________________

Address____

City _________________________________ State _________ Zip____________ Country ________

Telephone_____________________________ Fax ______________________ Email______________

Mail or fax this form to: Publishing Technology Management, Inc., 46 Lewis Avenue, Arlington MA 02174-3206
Fax: (617) 648-0678 • To order by phone call: (617) 643-8855

How To Find Out More About Companies Mentioned In This Issue

Adobe Systems, Inc.
1585 Charleston Road
Mountain View, CA 94039
(415) 961-4400

Alliance Technologies, Inc.
6034 West Courtyard Drive,
Suite 250
Austin, TX 78730
(800) 965-9857

Apple Computer
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010

Berger-Levrault/AIS
34, avenue du Roule
Neuilly-sur-Seine, 92200
France
+33 1 46 40 10 60
Fax +33 1 46 40 18 44

Boss Logic
1901 Landins Drive
Mountain View, CA 94043
(415) 903-7000

Documentum
5724 West Las Positas Blvd.
Pleasanton, CA 94588
(510) 460-4115

Electronic Book Technology
1 Richmond Square
Providence, RI 02906
(401) 421-9550

FileNet
3565 Harbor Blvd.
Costa Mesa, CA 92626
(714) 966-3400

Frame Technology
1010 Rincon Circle
San Jose, CA 95131
(408) 433-3311

Fulcrum
785 Carling Avenue
Ottawa, ON K1S 5H4
Canada

IBM Corporation
Old Orchard Road
Armonk, N.Y. 10504
(914) 765-1900

InfoDesign
100 Wellesley St. East,
Suite 100
Toronto, ON M4Y 1H5
Canada
(416) 928-6800

Interleaf
Prospect Place, 9 Hillside Ave.
Waltham, MA 02154
(617) 290-0710

Owl International, Inc.
2800 156th Avenue SE
Bellevue, WA 98007
(206) 747-3203

Information Dimensions, Inc.
655 Metro Place South
Dublin, OH 43017-1396
(800) 328-2648

Eastman Kodak Company
343 State Street
Rochester, NY 14650
(716) 724-4874

Micro Dynamics, Ltd.
8555 Sixteenth Street
Suite 701
Silver Spring, MD 20910
(301) 589-6300

Open Text Corporation
Suite 550,
180 King Street South
Waterloo, ON N2J 1P8
Canada
(519) 571-7111

WAIS, Inc.
1040 Noel Drive
Menlo Park, CA 94025
(415) 327-WAIS

Wang Laboratories, Inc.
One Industrial Avenue
Lowell, MA 01851
(508) 459-5000

Westinghouse Electric
Corporation
P.O. Box 746
Baltimore, MD 21298-9451
(800) 742-4802

Workgroup Solutions
76 Blanchard Rd.
Burlington, MA 01803
(617) 229-9000

Verity
1550 Plymouth
Mountain View, CA 94043
(415) 960-7600

Xyvision, Inc.
101 Edgewater Dr.
Wakefield, MA 01880
(617) 254-4100

The Gilbane Report on Open Information & Document Systems is published 6 times a year.

The Gilbane Report is an independent publication offering objective analysis of technology and business issues. The report does not provide advertising, product reviews, testing or vendor recommendations. We do discuss particular pieces of product technology that are appropriate to the topic under analysis, and welcome product information and input from vendors.

Letters to the editor are encouraged and will be answered. Mail to Editor, The Gilbane Report, Publishing Technology Management, Inc., 46 Lewis Avenue, Arlington, MA 02174-3206, or fgilbane@world.std.com or APPLELINK:PTM

ISSN 1067-8719