THE GILBANE REPORT™
on Open Information & Document Systems

Vol. 3, No. 2
May/June
1995

Publisher:
CAP Ventures
(617) 837-7200

Editorial Director:
Frank Gilbane
frank_gilbane@capv.com
(617) 547-2929
(617) 837-7200

Editor:
David Weinberger
self@evident.com
(617) 738-8323

Subscriptions:
Missy Mannai
missy_mannai@capv.com
(617) 837-7200

Design & Production:
Deborah Laite
(617) 837-0513

Authoring for Document Management — Beyond Word Processing

Most of our articles have focused on managing documents after they have been created. This is due to the rapid growth of and interest in compound document management systems, and the critical shortage of available information. Because of this growth and the growth of workgroup applications in general, there are evolving requirements for authoring systems. Collaborative authoring, authoring for document repositories, and authoring for multiple media delivery requires re-thinking existing authoring processes and strategies. In this issue we help you get started on the issues to consider.

InterDoc To Distribute Gilbane Report in Canada

We would like to welcome our new Canadian partner. InterDoc is a new Montreal-based company providing a suite of products and services to the Canadian document management market. One of their products will be The Gilbane Report. Our Canadian subscribers will continue to receive the complete contents of the U.S. edition, but with additional Canadian content provided by InterDoc, including Canadian news and editorials. InterDoc is also organizing, with the help of CAP Ventures and the Graphics Communications Association (GCA), a Documation Canada conference. Documation Canada will be held in Québec City, November 13-14. (See page 30 for further information).

Contents

Authoring for Document Management — Beyond Word Processing

Executive Summary

Strategic Overview

- There is a big difference between what's needed to make an individual author productive and to make a workgroup of authors productive.
- Word processors are reaching maturity in their ability to boost the productivity of individual authors.
- Authoring software for workgroups should be viewed as a vital constituent of a document management system.

Authoring for Document Management

- Authoring is a workgroup activity, not a standalone application.
- The basic strategy is to build a system that allows for collaboration, that structures documents and processes, and that deals with documents at a significant level of abstraction.
- The first step is to understand your basic document types. A document type is an abstraction that characterizes a set of documents according to their expected content, format, organization, tone and behavior.
- To accomplish content generation, you may have to trade off word processing functionality and features specific to your application.
- Although there is no longer any such thing as "the" WYSIWYG view of a document, for many authors WYSIWYGness is important because it helps highlight structure and they're familiar with it.
- When converting documents from other formats, try to preserve (or infer) structure.
- Look for tools for automatically incorporating information from databases.
- How much control you give your authors over the formatting of a document depends upon your workflow. In many instances, control has passed to professional document designers.
- Some authoring systems take a layout-based approach which gives better control over formatting. Others take a rule-based approach which affords more automation.
- Documents are highly complex and an authoring system ought to be able to understand and control their structure, even in multi-volume sets.
- Hyperlinking has become an important capability, providing an alternative web on top of a set of structured documents.
- Contents from multiple authors need to be normalized and composed into a continuous document set. The authoring system ought to handle all the "details," ensuring the integrity of the numbering streams, etc.
- Conditional text can let contents be shown or hidden based on attribute values and a control expression.
- Revision management can handle multiple drafts.
- A workgroup authoring system may offer some workflow capabilities as well, although these are frequently handled by the document management system.

- You want to find highly flexible and automated ways of outputting your documents to different media while ensuring their integrity and aesthetic qualities.

Structured Authoring and SGML

- Standard Generalized Markup Language (SGML) is the best way of capturing a document's structure and brings important benefits, making documents reusable, vendor-independent, and manageable down to the smallest degree of granularity.
- SGML is hard. You have to think through your document types, create document type definitions, and give your authors word processors that traditionally have been strong in SGML in inverse proportion to their strength as word processors.
- Although SGML authoring systems are better than ever, there are still two fundamental approaches: work in SGML or work in a word processor and convert to SGML.

The Future of Authoring

- More and more documents will be put on-line.
- Multimedia will be ever more important.
- Increasingly, the value of documents is in their links, not just their contents.
- Workgroups are getting larger, with more informal members.
- Increasingly, authors will work on fragments of documents, as the notion of "the" document is push aside by "repurposing."
- Bad HTML will drive out good SGML.

Risks and Costs

- Authoring seats cost more than word processing seats.
- The software can be harder to learn, and can be slower for lightweight tasks (but far faster for heavyweight document processing).
- Creating an authoring system is a type of application design, requiring distinct skills.
- There may be conversion costs.
- The market demands are changing rapidly so today's system may be obsolete in 18 months.

Conclusions and Recommendations

- Build an authoring system the way you would build an application: spec it out, look at your options from various vendors, and prepare for long term maintenance.
- It is not easy to create a functioning, efficient, and productive authoring workgroup, but the tools are in place for you to make tremendous progress.

Strategic Overview

If the 1980s were about empowering authors, the 1990s is about empowering workgroups, departments and divisions.

The generation of word processors we grew up with has become truly awesome at enabling power typing, making revisions instantaneous, and providing WYSIWYG formatting. These functions are not the same ones required to make an author a productive member of a workgroup, creating and maintaining documents over a network using a document management system.

For example, multi-level undo can be a lifesaver for a busy author, but authors in a document management system also need robust revision tracking. Individual authors may

need powerful formatting tools, but authors in a document management system may be required to leave their documents unformatted, but highly structured.

Exactly which features are required depends upon a highly complex set of factors, including:

- Is the document management system being imposed from the top, or is it being grown out of the current set of word processing tools?
- How controlled can the environment be? How frequently must documents from outside the strictures of the system be incorporated?
- What is the most efficient division of labor? Should authors become "paragraph grunts" who work on individual data chunks outside of the context of the final document? Or should they be given cradle-to-grave responsibility for the document as a whole?
- What is the life cycle of the document? How is it reused (repurposed)? How often is it revised?
- Are there government, industry or corporate requirements that must be met?
- How important is it that the system be standards-based?
- What are the human factors — resistance to change, pride in one's work, a sense of ownership, the sense of community?

Only by answering such questions can you decide what to look for in an authoring system. This article aims at helping you to sort through the issues.

Authoring for Document Management

Authoring is no longer a stand-alone application. Even when a single document is created by a single author, the document is inevitably part of a larger collection of documents that needs to be managed, found, retrieved and reused. This changes the requirements of authoring systems.

The Managed Environment

Documents within an environment that has (or perhaps just needs) a document management system typically are characterized as follows:

- High-value with a significant half-life (*i.e.*, stays valuable for a long time)
- Related to other documents by topic — either as multipart documents or as parts of a unified collection of documents
- Go through a controlled review and revision cycle
- May be "repurposed," *i.e.*, written once but published in a variety of forms and formats
- May have to meet rigorous standards for content, structure, and format

In a document management system, we expect to find many of the following features:

- Check-in and check-out of documents (and perhaps document elements or fragments) from a centralized library
- Centralized control over access for reading, writing, and commenting on documents
- A way to search for documents
- Revision tracking
- A way to view documents without using the original authoring application
- Some type of configuration management so that the dependencies among documents are tracked

- Some type of workflow management so that documents are automatically routed to the appropriate people and tasks throughout their life cycle

In such an environment, a single document is like a single bee; to understand any of the parts, one has to consider the real organism to be the hive itself.

Basic Strategy

"The three keys to successfully authoring within a document management system are collaboration, structure and abstraction."

The three keys to successfully authoring within a document management system are collaboration, structure, and abstraction.

Collaboration means enabling authors and the rest of those involved in creating and reviewing documents to communicate efficiently. To do this, people need to be able to find, view, and comment on one another's work. At a minimum, it requires straightening out the predictable document and application incompatibilities so that the graphic artists on the second floor don't have to go through a fire drill every time they want the marketing group on the third floor to look at some suggested artwork, and so the technical authors on the fourth floor don't have to run to the fifth floor to find a machine capable of reading their floppies so they can convert their chapter into a format that can be understood by managers on the sixth floor.

Collaboration means more than removing the obstacles so that data can be passed around. It also means using networked technology to enable a community to engage in the day to day conversation that results in new ideas and the polishing of current work. In this article we will discuss the issues around the interactions typical of the authoring process, such as comment and approval processes and the processes by which the work of multiple authors is combined into one unified volume.

Structure — the next key — enables efficient collaboration. By structuring your documents, you make them easier to manage and easier for recipients to use. By structuring your processes, you gain control and efficiency. Structure may mean moving to SGML, but it may also mean insisting on using styles in Microsoft Word documents. It may mean installing a robust workflow management system or it may mean distributing paper schedules. No matter how much structure is required, the bumper sticker on a workgroup author's car ought to be "Structured and Proud."

The third key is **abstraction**. Word processors beat typewriters in part because the typewriter is a totally non-abstract, concrete authoring tool: you get no view of the document other than that of "what character comes next?" With authoring systems, you can escape from this view to think about the document as, for example, something with a structure. You can think of a subhead not just as centered text but as a type of element defined by its role in the document not just by its page position. You can think of how the pieces of documents relate and may be reused.

That is one concrete advantage of abstraction: you can design a system so that you get maximum reuse of the work that you're doing. Through abstraction of format, content, and structure you can create elements that can be "repurposed" into multiple documents by applying different formatting rules to them. For example, a product description may be written once and used in a glossy catalog, a faxable product data sheet, a black and white training manual and an on-line reference manual. The second bumper sticker for a workgroup author ought to be "Abstract writers do it at a higher level."

The Stages of Development

In the next sections we will look at the various stages of the development of documents within the "hive" context of document management. Repeatedly, we will see the role of collaboration, structure, and abstraction.

Document Type Design

Authors outside of a document management system make use of templates. Within a document management system, document types are more important than templates.

"Document types are more important than templates."

Templates are documents that contain frequently used document elements already formatted according to some standard. A document type — a phrase popularized by SGML, which has had a great deal of influence in this field — is more abstract than a template. A template is always embodied in a particular file which gets copied and used. A document type may exist first and foremost as a description of a document, although it will inevitably be embodied in documents as well.

A document type specifies what is expected (or required) of a document of a particular sort. It recognizes that documents fall into recognizable types (or "classes," as the object-oriented folk would say) memos, reports, white papers, laundry lists, ransom notes, fortune cookie fortunes, etc. A document of a particular types carries with it certain expectations, including:

- Content — what text and graphics are expected in it
- Format — what it looks like
- Organization — the order of the parts
- Tone — what it sounds like
- Behavior — what you are to do with it

Different systems allow different sets of these expectations to be defined in a document type. For example, the system may provide a way to express the acceptable elements in a document, their acceptable order, and their acceptable formatting.

Systems also vary in how they express these expectations. Some use basic templates that focus on getting the elements formatted correctly. Others express the expectations as rules that require the author to create a document that meets requirements. These rules themselves can be expressed rigidly or flexibly, during or after the editing process. For example, a system may not enable an author to put a by-line element anywhere except after a title. Another may have the same inflexibility (desirable in some instances) but will only check the structure of the document when the author saves the document.

Since SGML is the ultimate way of working with document types, you will find a further discussion of these issues in the SGML section below.

Content Creation

The first step is to create the document contents. We will not discuss the requirements for a competent word processing system since word processing is the single most prevalent application on computers, and because *The Gilbane Report* in any case does not discuss religious issues.

We will instead, point to a couple of areas in which you have to make some decisions.

Word Processing vs. Application-Specific Functionality

There is a balance one frequently has to strike between word processing functionality and other types of functionality. For example, there are authoring systems that are extremely useful in creating documents destined for the Internet's World Wide Web. They are, however, quite under-featured in terms of word processing when compared with Microsoft Word or Novell WordPerfect. You must make the decision about which route to go (or to adopt a hybrid approach, using a word processor for generating text and a conversion process to change it into a Web document) based upon your needs.

WYSIWYG vs. Content Purity

Typical desktop word processors expect authors to work in a WYSIWYG view of documents, although they may also afford draft mode and outline views. WYSIWYG can lose its value — and, to some degree, its sense — in authoring systems within document management systems. If an author is contributing to a larger whole, and if the design of the document is left to specialists later in the process, there is little value in having the author worry about what the look of the document should be. And, if the author is creating contents that will be "repurposed" for use in a marketing brochure, a training manual, and an on-line service manual, there is no such thing as "the" WYSIWYG view of the document.

WYSIWYG is not a mere frill, however. First, it provides humans with a very fast way of understanding the structure of a document: heads are bigger and centered, and footnotes are at the bottom of the page. This can help an authors work in a structured way and can help them spot mistakes in the structure. (See Figure 1) Second, moving from a WYSIWYG word processor on your desktop at home to a non-WYSIWYG system on your more powerful system at work can feel like an unnecessary degradation of quality and even of prestige.

Fortunately, most systems that do not present WYSIWYG as a primary authorial view present something like a WYSIWYG way of working that at least uses font changes to indicate various elements of structure, even if what's on the author's screen is not what will be finally output to the reader.

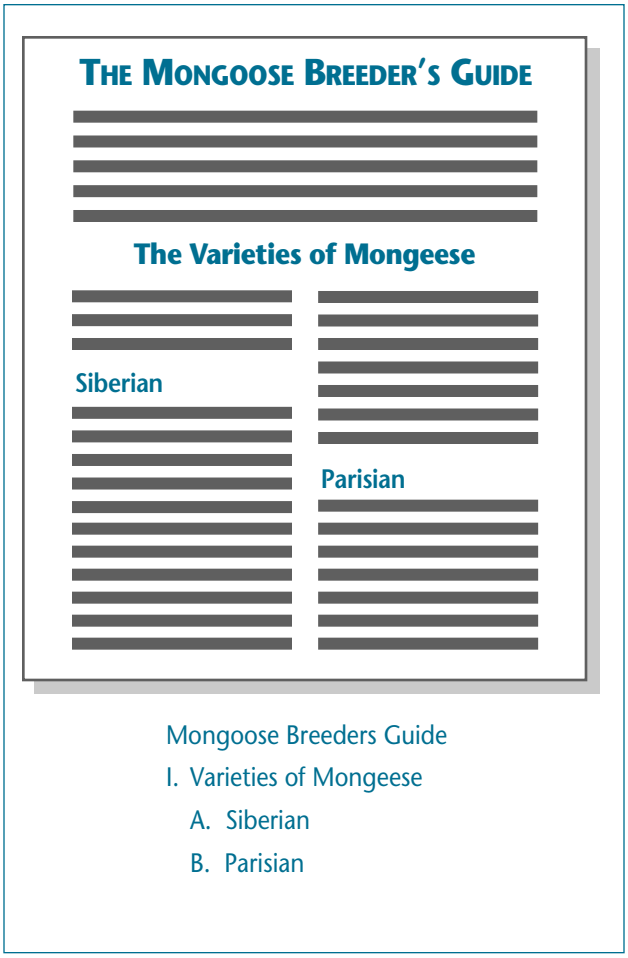


Figure 1 WYSIWYG view provides "intuitive" way of seeing a document's structure.

Document Conversion

Another source of contents is work that already exists. This may be legacy work or work coming in to the organization from an outside source such as a contract writer or a component supplier.

Frequently, this will mean converting the contents into a data format that can be used well. What does it mean to use a format well? You can optimize to preserve some, but probably not all, the information in the original document: contents, structure, format, graphics, etc.

We recommend, of course, that you optimize for preserving structure. In fact, there is software that infers structure from formatting information, so that it will figure that a bold line of text is probably some type of subhead, This is one of the few ways in which a filter or converter can actually improve the quality of your data.

It is also important that the processes by which you convert and absorb foreign data is a well-understood process. If an author has just spent three hours tracking down a filter that will convert *foo* documents into *bar* documents, there is no reason why anyone else should ever have to go through the same time-wasting effort. Build a known library of filters. Consider installing a PC — last year's model is more than powerful enough — in a public place to act as a converter server so that the unlucky person who has the PC with the foo-to-bar conversion software doesn't have to be interrupted whenever some filtering needs to be done.

Database Integration

There is yet another source of information that often needs to be incorporated into the contents of documents created by workgroup authors: data in a database. The objective is to automate the process by which this data is assembled and brought into the document.

For occasional or *ad hoc* queries, using the database management system's ability to generate reports may be sufficient; print them to a file and stick them into your document. If, however, this is a repeated and predictable part of your work, then it should be automated. Most authoring packages provide some facility for this, or a way in which to attach an "extension" that will do the job. Look for:

- The ability to embed queries in the document template so that they can be automatically executed
- The ability to take advantage of the power of database management systems, such as performing joins, handling repeating lists, etc.
- An easy to use interface that hides the complexity of SQL (or other query language) from the author

Multimedia Authoring

As multimedia applications have arisen, so have authoring systems designed specifically for such applications. These applications tend to present a very different model of a document, conceiving them not as pages with text to be filled in but as applications with buttons to activate. They work on the notion of the screen, rather than that of the page. They provide tools for putting in buttons and "hot spots" that allow for user interactivity. And they include ways in which to coordinate the timing of events.

There is a distinction — being overcome as the technology advances and expands — between systems that focus on building multimedia applications such as games and those that focus on building multimedia apps that replace traditional documents such as training manuals. The latter, of course, provide more in the way of traditional authoring and structuring tools.

Word processors and standard authoring tools are also beginning to add multimedia capabilities, some through built-in capabilities and others by providing toolkits for expanding the software.

Increasingly, authors are going to have to supply content that not only gets published on paper or on-line, but which will be repurposed into multimedia apps. Initially the multimedia apps will be like paper apps with some buttons added and the contents will be successfully repurposed. Increasingly, however, we believe that multimedia apps will be sufficiently original (and appropriate to their media) that contents designed for documents will not be useful for the multimedia expression of the same data; at that point, the content author's work will be used as an information source for a multimedia application designer, much as engineering specs are frequently the source for technical documentation.

Formatting

How much control you want to give your authors over the look of the document has entirely to do with the work process you have established. If the document is to be laid out by designers after the authors have generated the content, then clearly you do not want to have your authors waste their time applying formats that will then be undone by the designers. If, however, the documents are simple enough in their format that they can go from the author to print with little cleanup, then you may want to give the authors more control.

In either case, it is vital that the authors be able to indicate the structural relationships that will be reflected in the format.

Authoring systems take two basic approaches. The layout-based approach allows authors to draw boxes for structural elements which then size themselves to contents. The rules-based approach allows the author to specify the parameters for document layout which the system then applies automatically. The layout-based approach gives greater hand control over the placement of elements on pages. The rules-based approach allows for greater automation when applying or modifying layouts in large, complex or multiple documents.

The most basic issue between these two approaches is that of revisability. As you are given more control over the hand manipulation of the layout of individual pages, the aesthetic quality of your documents goes up but their revisability goes down. That is, it's easier for complex revisions to break a highly-designed page.

Some layout features to look for. (See Figure 2)

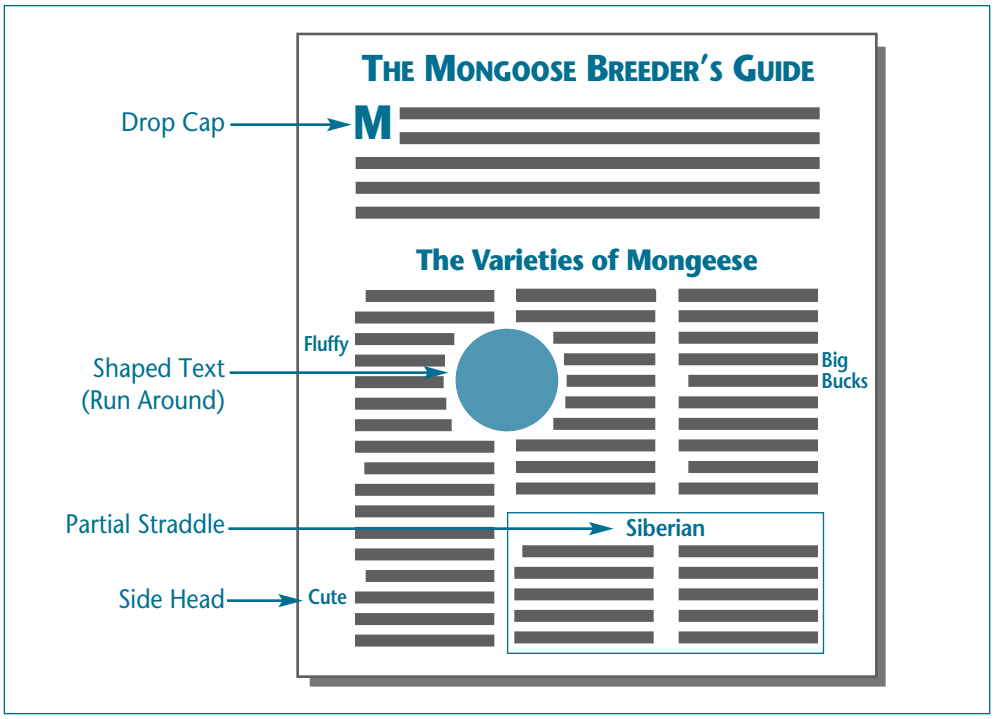


Figure 2 Some layout capabilities.

- Multiple column control, allowing section heads to straddle all or some columns.
- Text shaping so that text blocks can have the margins of their individual lines set. This can be done automatically by placing the text over a shape or by hand. The system may shape the text itself (so that as it moves over the pages the paragraphs retain their shape) or create a "shape zone" on the page so that any text that moves over it is

shaped. (For the former, notice what happens when the paragraphs break across pages. For the latter, consider the risk in tying a shape to particular page when large-scale changes may move inappropriate material over that page.)

- Side heads so that a section can be labeled with some text parallel to its beginning in the outer margin. Make sure the side heads move with the text it is labeling.
- Graphic control. Frames should be capable of being tied to text in sophisticated ways so that they stay with the text they illustrate and are positioned in aesthetically pleasing

Three Document Architectures

Historically, there have been three ways in which authoring systems have approached documents as a data structure. (See Figure 3)

Procedural markup inserts formatting codes into the stream of ASCII characters. For example, ".pn" as the first characters on a line might indicate a page break. These codes are interpreted by the word processor and used to format the document. Initially, the author inserted these codes by typing them in directly and they would be used to format the document only when it went to print it. Now they are rendered on screen as well.

Declarative markup inserts tags naming paragraphs and other structural elements. A header or separate file defines the formatting of the tag names. (Procedural markup may also be allowed so that non-structural strings — such as an arbitrary sequence of letters — can be formatted with italics, etc.) Declarative markup abstracts content, structure and format so that new formats can be applied simply by altering the definitions in the header. An element can even have content — an autonumber or initial graphic, for example — as part of its definition.

Object orientation, like declarative markup, tags elements of structure and defines their format external to the elements themselves. However, an object oriented system has a class structure, so that any object is an instance of a class which itself may be a sub-class of a higher class. A class can define not only format but also behaviors so that, for example, a fill-in class might include range-checking of contents typed into it.

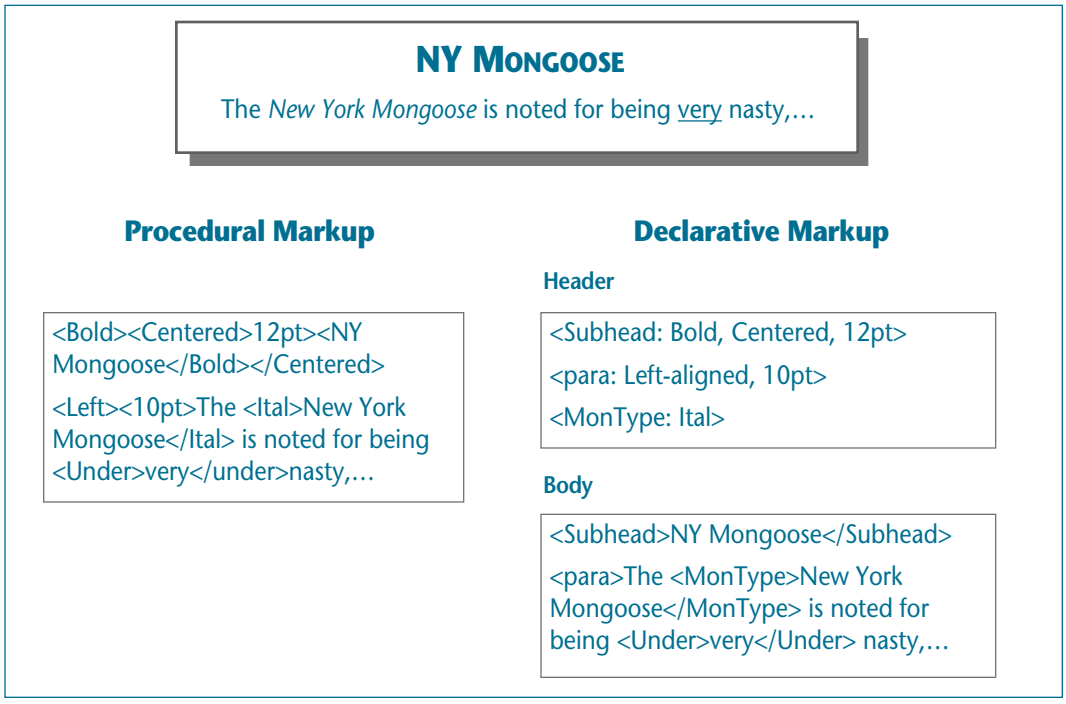


Figure 3 Simplified example of procedural and declarative markup.

ways by the software. They should also be capable of being placed under or on top of other elements in the document stream.

- Mixed page shapes so that a landscape page can be inserted into a portrait document.

Systems generally are able to let you hand tweak pages to just about any effect you want. The question is the trade-off you will pay between ease of use and revisability.

Structuring the Document

For authoring within a document management system, nothing is more important than structuring the document. There are two basic questions: How much structure does the system understand, and how well does it allow the author to interact with and control the structure?

"There's a strong case to be made that documents are the most complex structures handled by computers."

There's a strong case to be made that documents are the most complex structures handled by computers. There is an equally strong case to be made that no system is yet able to handle the full complexity of documents; the most rigorous systems succeed only by arbitrarily limiting the complexity of documents.

One level of complexity has to do with the hierarchical nature of documents. At a minimum, a structured authoring system has to provide a way of identifying individual structural elements such as paragraphs and headers, but also should include "inline" elements such as embedded heads and product numbers. Elements, however, are embedded within larger units such as subsections which are themselves part of sections, chapters, documents and volumes. Understanding and sensibly maintaining the integrity of such complex structures is very difficult.

This type of structure can be represented as an outline or tree. Documents frequently have elements that are not as obviously represented in an outline. For instance, there are footnotes, endnotes, sidebars, call outs, and hyperlinks.

In addition to these levels of structure, there may be a class structure representing the relationships among the various element definitions.

There may also be rules about permissible structure.

Then there are document-specific ways of expressing structure and making it apparent to the reader. For example, structure is often cued by fonts, page placement (top of the page may signify that an element is higher in the pecking order) and document placement (end of the document may indicate elements more loosely attached, such as appendices).

One common way of expressing structure is through number streams. Looking at how flexible automatic numbering is can give you one indication of how sophisticated an authoring package is at handling structure. You want to see if the system can handle multiple numbering streams that can be continuous, interrupted and restarted, and which can be used in and out of graphic elements. Then you want to see if automatically-placed numbers can be cross referenced automatically, across documents in a multi-document set.

So, structure is complex and requires sophisticated tools for expressing it.

Equally important, of course, are the tools the system provides to the author for manipulating the structure.

First the author needs to be able to see the structure. Of course the author needs to be able to see the names of the individual elements; some systems show the name of the current element, and others list the element names in a bar to the left of the document window. If name labels are shown for all visible elements, it is useful to be able to manip-

ulate the document by manipulating the labels, selecting them (perhaps discontinuously), moving them, and applying format changes to them.

(An important point: Structured systems let you modify the formats of all instances of a class of element by modifying the class definition. But check to see how flexible the system is. Can you modify one particular instance, apply changes globally to all instances, and have the modified instance maintain its change? For example, suppose you indent one instance of "body text" and then decide to change the font for all instances of "body text." You likely want the indented instance to maintain its indent.)

A simple outline view helps a great deal. So does the ability to see relationships that outlines obscure, such as the dependency of graphics and hyperlinks. In addition, it can be useful to see other hierarchies (such as the tree of class relations) and to be able to generate various types of reports (how many elements, which classes of elements are used, are there any unlinked hyperlinks, etc.) (See Figure 4)

Figure 4 Hyperlinks (dotted lines) can create a new web layered on top of a set of well-structured documents.

"Workgroups also need to pay very careful attention to making sure that all contributors structure their work in a common way."

Workgroups need to pay particular attention to the possibility of building structured documents within multi document sets. The structure may be as strict as that of chapters in a book or as loose as supporting material for a proposal. In fact, there may be multiple structures in a complex, multi-document set. For example, an encyclopedia has a straightforward hierarchical structure, based on alphabetical order at the top level. But it also has a web of cross references which enable people to find information outside of the alphabetical order. The web — typical of hyperlinked documents — is as demanding a structure as the more formal top-level one. (See section on hyperlink authoring below.)

Workgroups also need to pay very careful attention to making sure that all contributors structure their work in a common way. For example, all the authors of a technical reference need to know from the outset that every chapter will begin with an exploded view of the functional area under discussion and will discuss sub-assemblies based on the order in which the repairperson encounters them. Further, they need to know that every repair procedure description has to begin with a list of tools, and that the authors have three levels of warnings they can use. That is, they need to know the "tag set," the structure and the "content model." (See the section on SGML for more detail.)

In short, designing a suitable document type, propagating it and ensuring it is followed are all crucial to managing a workgroup of authors successfully.

Hyperlink Authoring

Adding hyperlinks to a document has become a necessity as documents are increasingly being published on-line. An authoring system should make it as easy as point-and-link to create a hyperlink. It also should be able to maintain the links throughout the revision process — having to re-link links by hand can be as time consuming as creating them in the first place. In addition, the system may enable you to set some options for links, such as linking words, providing buttons, or pop-up lists to offer the user choices. And of course, the system should be able to output the document automatically in HTML.

Composing Complex Docs

By now you have generated contents within a well-structured document. The next stage (which may in fact be concurrent with the rest) is to compose these contents into a coordinated document set.

Part of this requires the system to lay out the document elements. In some workflows this will be an entirely separate part of the process, accomplished by designers apart from authors. In other workflows, this is accomplished with the authoring software itself.

In either case, the contents have to be normalized. For example, any differences between tag sets (or style names) has to be noticed and resolved. In addition, streams of information that flow from one document to another need to be resolved as well. For example, some automatic numbering streams need to be restarted for each chapter and some need to be continuous, and indexes and tables of contents have to be maintained appropriately.

Systems vary widely in how they enable a workgroup to build a multi-document document. Some provide an outline view that shows the set of documents in the master volume. Others provide a graphical user interface with icons representing books and documents. In addition, systems vary in how dynamically they update the information.

The important thing is to remove from the author the need to worry about the state of the overall document based upon what the other authors have done. For example, you don't want the author of chapter five to have to call up the author of chapter four to find out what the last automatically numbered figure is in chapter four so that they know where to start numbering the figures in chapter five. It should be up to the system to

maintain the integrity of the overall document, and to do so in ways that are unobtrusive and utterly reliable.

Conditional Text

Some authoring systems enable you to include or exclude particular elements programmatically so that you can on the fly, create new versions of documents with varying content.

At the simplest, you can force individual elements in or out without having to delete them. You may also be able to show or hide an entire class of objects, *e.g.*, diagrams or side notes. Some systems allow you to show or hide elements based not only on their tag names but also on attributes that you have attached. For example, you might create an attribute called "level" and give different elements different values for it; an element covering a beginner's topic might have a value of one, and a very advanced topic might have a value of five. Then you could create the advanced version of a document by hiding any element whose "level" attribute has a value less than three. Finally, some systems enable you to create complex Boolean expressions to control the contents of a document, such as "Show me only those elements whose level attribute is greater than three and whose platform element is either PC or Mac but not UNIX." Change the control expression and you change the exhibited contents.

The benefit of using conditional text is that you can produce multiple versions of a document from a single source. The price you pay, however, is in the preparatory work that has to be done to tag the elements with appropriate values.

Extending Authoring Systems

Building a workgroup authoring system really is building an application. You may not have to do any programming, but you do have to think through all the functionality and processes you need, and then build the system that comes closest to your needs. You will adjust the software's options and profiles, and will build templates (or document type definitions) for your group.

Sometimes, the software just doesn't do what you need it to do. It doesn't have functionality you need, or you need to integrate several pieces as smoothly as possible.

That's where extension environments come in.

Many systems provide an open API so that you can integrate them with other applications. An API allows an external application to "call" the functionality built into the first application without having to bother a human being. This can also enable a development environment such as Visual Basic to invoke the app's functionality so that you can build a new, customized user interface to do what you need. (Of course, you do this only when interacting with the app in batch mode is appropriate; you cannot rebuild a complete word processing UI using Visual Basic or some other development environment.)

Other systems provide an internal development environment that enables you to extend the functionality within the application. For example, you might be able to change the way the spell checker works so that it builds a list of misspelled words you can examine later on, or you might be able to create a multiple-choice element that provides a drop-down list of possible fill-in choices.

Creating customized or extended versions of the software can require the skills of a professional developer, but can also have a very dramatic payoff if your needs are severe enough.

Revision Management

Workgroups generally contain at least one person who lacks Mozart's ability to write a work that requires no revision. You can fire that person or you can look at the ability of the authoring software to track and manage revisions.

The basic revision capability allows you to declare that you are beginning a new version (or draft or edition — there is no agreement yet on the terms). Any text you add or delete is now shown in modified font; typically, additions are shown as underlines and deletions as strikethroughs. The system will likely add revision bars to indicate altered sections as well. (See Figure 5)

Any text you add is shown in modified font; typically, additions are ~~underscored~~ underlined and ~~cuts~~ deletions are shown in strikethrough. The system will likely add revisions bars to indicate altered sections as well.

Figure 5 Revisions frequently are shown with underlines, strike throughs and revision bars.

This capability can be extended in two directions. First, the system may let you track more than one level of revision. For example, when you declare a new draft for the second time, the changes in the first draft can still be marked as changes. This enables you to "roll back the clock" to see previous drafts. Second, the system may let you incorporate simultaneous editions. For example, you can take one base document, do one set of revisions to turn it into, say, a report that meets regulatory requirements for Wisconsin and do another set that make it suitable for Wyoming. The Wyoming version may itself then

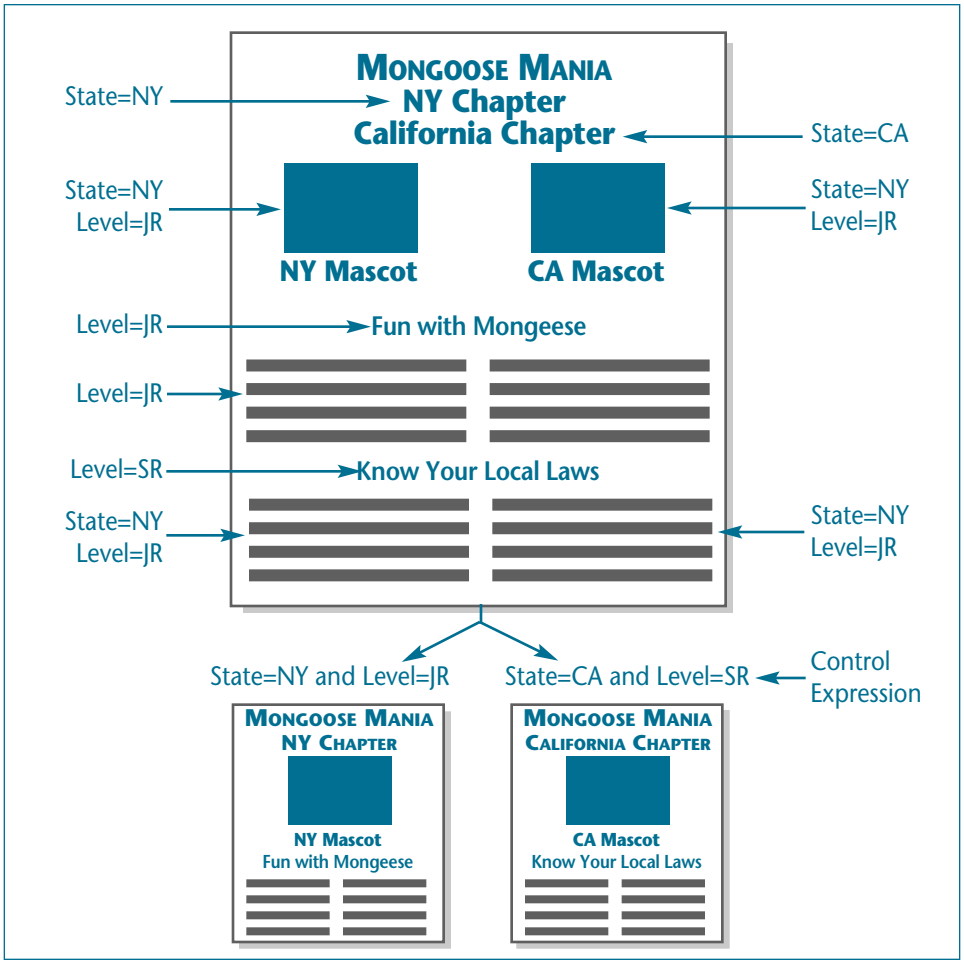


Figure 6 You can quickly assemble multiple versions by using conditional text.

go through four or five further drafts. This has the advantage of enabling you to alter the initial root document and have the changes propagated automatically throughout all of its "children."

A workgroup authoring system may also track additional information about elements, other than that they have been added or deleted. For example, it may also track who the author was, who edited it, and why the changes were made. This can be a very useful feature for workgroups. (See Figure 6 on previous page)

If the authoring system provides revision control, this needs to be seamlessly integrated with the versioning provided by every document management system. If the authoring system creates revisions by maintaining changes in a single document, then the workflow can provide macro versioning and the authoring system can provide micro versioning (so that the management system's version 7 may be an authoring system's document that itself contains three revision cycles). If both systems generate a new document for every revision cycle, then the integration of the two becomes more complete.

Workflow

Although workflow is an important part of a document management system, authoring systems may have some elements of workflow management. We have already discussed some areas that might be part of a workflow system, including revision tracking, review and complex document management.

Desktop Word Processor vs. Industrial Authoring Systems

There used to be a clear distinction between desktop word processors and industrial strength authoring systems — a distinction as clear as DOS vs. UNIX. Now the distinctions have blurred, but there still are differences

- Size. Authoring systems should be able to handle documents hundreds of pages long without straining.
- Complexity. Look for more control and flexibility when dealing with complex structures. For example, look at how the systems manage autonumbering streams, and test drive complex multi-page tables.
- Multi document control. Authoring systems offer more control and more robust features for workgroups working on documents that consist of many chapters.
- Repurposing. Through the abstraction of format, content and structure, and through mechanisms such as conditional text, an authoring system makes it easier to generate multiple documents from a single set of contents.
- Revision management. Robust, multi-level revision management is tough to do well. Authoring systems tend to be better at it than desktop word processors which view it as "undo" on steroids. (New versions of desktop word processors are beginning to offer impressive versions of this feature.)
- Extensibility. Because authoring systems are almost always part of a larger application, the vendors have provided more comprehensive APIs and extension environments.
- Cost. One very noticeable difference is that authoring systems are likely to cost more than desktop word processors.
- Performance. Word processors may give better performance on some light weight tasks, but authoring systems give far better performance when it comes to doing major document processing.

In addition, authoring systems can provide some degree of routing, so that a document is distributed to a set of reviewers or collaborators. This can be manual or automatic, through e-mail or by copying it over a network.

The key factor is how well you will be able to integrate the authoring system into your workflow management system (which very likely will be part of your overall document management system). The promulgation of the ODMA standard promises to make this much easier by enabling most authoring systems to put a "check in" option on their file menu.

Outputting It

Having done all this work, you might eventually want to put your document into the hands of readers. That means outputting it, either on paper or electronically.

In many instances, printing on paper will be as simple as pressing a button. There are some advanced capabilities worth looking for, however.

Depending on your workflow, you may want to be able to drive a demand printer from your authoring system. This requires some form of "job ticketing" so that the author can specify the options the demand printer can handle such as what sort of cover stock to use, inserting tabbed sheets between sections, and binding options.

If you are outputting it electronically (on CD-ROM or for Internet delivery), you need tools that enable you to:

- Create and check links
- Format it for on-line
- Generate interactive indexes and tables of contents
- Convert cross references to links
- Check to make sure it meets the medium's standards

Authoring for on-line distribution is a topic deserving its own treatment covered in a future issue of *The Gilbane Report*.

Structured Authoring and SGML

Structure is good. SGML is the best way of capturing a document's structure. The question is whether it's the best way of interacting with structure.

SGML has tremendous benefits for authors working within a document management system:

- It breaks documents into manageable parts
- It can assure that a document meets its specifications for organization and content
- It can make documents and their parts eminently reusable in other documents
- It enables documents to be exchanged with other SGML-competent systems
- It ensures that a document's contents and structure are safe even if you change authoring or document management systems

Why hasn't everyone adopted SGML?

Because it's harder than not doing SGML.

Creating an SGML application requires carefully thinking through all the document types you need to use. Then creating the document type definitions that will enable your authors to create those documents.

Once you've done that very important bit of document data modeling, you now need to give your authors a system for creating SGML documents. The initial SGML authoring

systems did not look or feel like ordinary word processors. They made their SGMLness very obvious. They did not have the word processing power we have come to expect. Those who adopted the SGML authoring systems did so because they needed the structure and discipline SGML provides.

Since those early days, the state of the art has advanced.

SGML authoring systems now provide considerable word processing power (usually not as much as the leading word processors). They let authors work in a view that exposes the SGML markup or hides it with WYSIWYG fonts. They provide more authoring tools that take advantage of the structure that SGML exposes, such as intelligent outline views.

A fundamental chasm has opened up between two approaches. One approach extends the SGML authoring tools so that they are easier to use. The other adds an SGML layer to an existing authoring tool; this layer may be integrated into the product dynamically so that SGML functionality is given to the author, or may be an intelligent conversion process performed at the end of an editing cycle. Neither approach has yet broken the market open to make SGML as prevalent as it deserves to be.

Three factors may help speed the proliferation of SGML. First, HTML's popularity is helping to educate the market about SGML (since HTML is an SGML application). Second, the major desktop word processors have recently added SGML capabilities, making SGML achievable by the mass market. Third, this year a number of SGML-based document management systems are becoming available, adding to the value of SGML documents.

But, whether the rest of the world uses SGML or not, you need to decide if an SGML authoring system is the way to go. Here are some considerations:

- How predictable are the document types that you deal with? If you often invent new types that you use once or twice and discard, it is unlikely that the effort required to create a new Document Type Definition (DTD) will be worth it.
- What is your workflow? If your authors are involved in publishing tasks such as layout, then SGML is a poor choice. On the other hand, if your workflow separates content generation from formatting, then SGML may make a lot of sense.
- Do you do a lot of repurposing? If you want to build a library of content modules that can be easily assembled into new documents, nothing beats SGML.
- Are there contractual or governmental regulations requiring you to create SGML? If so, end of story.

The Future of Authoring

There are some trends already developing that will affect the future of authoring:

- More and more documents will be delivered electronically, particularly reference works where the reader gets the benefit of computerized access and navigation.
- With electronic delivery, multimedia becomes more and more important
- With the growth of the World Wide Web, the value of documents increasingly is in their being intelligent nodes — *i.e.*, linked — to other documents. Linking well will become almost as important as writing well.
- More and more authors will be brought into the workgroup as extended family members. Since there won't be any uniform standard, authoring systems are going to have to continue to improve at assimilating foreign material.
- The concept of "the" document will wane as documents are generated on the fly from databases of contents. To the end user they will look and feel like documents, but the

authors will work on the content fragments, and will feel like paragraph jockeys abstracted from "the" WYSIWYG document.

- A more controversial prediction: Bad HTML will drive out good SGML. SGML will be established as a standard not from the top down by a world that looks upon SGML's shining countenance and decides it is good. Rather, SGML will be established as a standard by a world that embraces HTML and then turns to SGML to extend it when HTML's shortcomings become obvious.

Risks and Costs

What are the risks and costs of adopting an authoring system designed to work within a document management system?

- The individual software seats are more expensive and often require more hardware investment.
- The software can be harder to learn because of its increased functionality.
- The software may not perform as well for lightweight tasks, although it should perform far better on heavy weight document processing chores.
- Creating an authoring system requires the skills and patience involved in creating any multi-user application.
- Committing to one authoring system and then changing to another involves conversion costs and data loss, the extent to which depends upon which systems you've chosen. (SGML systems have entirely transferable content and structure data, but there still may be costs for converting to new formatters.)
- The market demands are changing rapidly — who would have predicted that HTML output would be a *sine qua non* 18 months ago? — today's perfect system may be inadequate tomorrow. This can be ameliorated by choosing a system that is highly extensible.

Recommendations

If you are thinking of using an authoring system that is part of a document management system, you are talking about building an application. This requires all the usual work involved in building an application: create specifications, build a data model, investigate software options, prepare for long-term maintenance, etc.

- Gather your documents and divide them into types. Consider what has to go into each, how much flexibility you need, etc.
- Look at your inputs. Are your documents' contents coming from databases, other vendors, e-mail, hallway conversations? You will need to smooth the entry for each and every data source.
- It is very important that you think not only about what individual authors do but how the workgroup as a whole will operate. What are the current steps in the development of workgroup documents? What is required to make the collaboration smoother?
- Make a list of the features you want and then put it aside. Gather information from all the vendors you can about the features they offer. The good news is that you are likely to find useful features that you had not put on your list because the vendors are in the full-time business of coming up with useful tools for users. Make a new list of features you need.
- Do not rely on vendors' features lists. There is a big difference between having a feature and having a useable feature. For example, every vendor has automatic numbering, but

you won't really be able to tell which one does what you want unless and until you load the software and give it a thorough workout in realistic circumstances.

- Consider your culture. It may turn out to be less disruptive and expensive to let your authors have the illusion of WYSIWYG editing even if in the next step down the road all the formatting information is stripped away. On the other hand, SGML authoring vendors report that it is very common for authors new to their system to start out in WYSIWYG mode and within a few weeks to prefer to work in show-all-tags mode.

In conclusion: The problem of putting text on paper was solved a long time ago. The challenge of creating a functioning, efficient, productive authoring workgroup is still real. With today's tools, you can make tremendous strides by remembering to consider collaboration, structure, and abstraction.

David Weinberger

OLE AND SGML

Introduction

In our continuing attempt to make sense of the components of document management, here are two articles that untangle some of the threads of SGML and Object Linking and Embedding (OLE), with object-orientation lurking not far behind the scenes. The first is from an application perspective by the editor of this newsletter, David Weinberger. The second takes an architectural look; it is contributed by Michael Cockrill who, as program manager for Microsoft Word SGML Author is uniquely well-qualified to discuss the relationship of the two standards.

An Application Perspective

The basic issue that makes the relation of SGML and OLE a confusing topic is that — like capitalism and democracy — they address fundamentally different concerns, and yet are global so that they are found operating in the same domain.

SGML basically is a set of rules for describing a document. In fact, it's a set of rules that lets you specify a set of rules for describing a document. SGML was designed around the notion that a useful document description will include all of the text contents, attributes attached to those contents, and the structure of the document. If you want to describe more than that, you can specify how you will describe other elements. For example, if you want to describe a table of information, you can establish the rules or model for doing so, and various models have been adopted as semi-standards by different industries.

Microsoft's OLE, on the other hand, is a way of sharing data and functionality — objects — among applications. With OLE, an object created in one application can be embedded into another, retaining a link to the originating application. (It can also be embedded as a severed copy; that's why OLE originally stood for "Object Linking and Embedding.") SGML is a set of rules for how data that is essentially static (a snapshot of a document) is to be represented, whereas OLE is a set of rules for how to communicate dynamic data.

SGML and OLE meet — or at least hang around together — in object-orientation. At a gross level, the marks of object-orientation are that the "stuff" divides into irregular chunks of data (and not into neat, equal-length fields and rows), and a chunk includes things it knows how to do (methods), not just traditional data. For example, in an object-oriented authoring system, the chunks might be paragraphs and graphics of various types and the methods might be things like "spell-check myself," "kern myself," or "rotate myself." In addition, object systems are arranged into classes and subclasses of objects; for documents, classes might be things such as text objects and graphic objects, and subclasses might be paragraph objects and technical drawings.

OLE is purely object-oriented. It is based on Microsoft's Common Object Model, and exhibits all the signs of classic object-orientation: an OLE object has data and methods, a class structure and inheritance (*i.e.*, if it doesn't have a method requested by some calling application, it looks to its parent for the method). An application capable of serving up OLE objects bundles in a set of capabilities any recipient app can activate; most OLE objects, for example, can display themselves using a Windows Metafile graphic packed into the bundle. Other capabilities may include an object allowing itself to be edited or to retrieve more information from the originating app (*e.g.*, it may be capable of taking a query in and outputting results).

SGML lends itself well to object-orientation. An SGML document neatly segments into chunks of data (elements) such as paragraphs, sections, and chapters, as well as "inline"

elements such as part numbers. An SGML document thus has the irregular data chunks typical of object-orientation. But SGML is at a higher level of abstraction; its elements don't include methods, but may have methods applied to them by SGML-cognizant applications. For example, "format thyself" methods may be added to an SGML document by an SGML document display application. So, although SGML is not object-oriented in itself, because of the nature of its data, and its high level of abstraction that allows methods to be applied to those objects, SGML authoring systems have tended to adopt an object-oriented approach, and SGML data management systems have also used an object-oriented data model.

"So, how do SGML and OLE relate? First, they clearly do not compete."

So, how do SGML and OLE relate?

First, they clearly do not compete. Although OLE has a way of expressing data hierarchically (OLE structured storage), in theory, you could use it to express document structure without using SGML, you would lose the cross-platform, vendor neutral benefits of SGML without gaining anything. Let's just forget I mentioned it.

In a more realistic application, an SGML document in an SGML authoring system might accept OLE objects. You could drag and drop a chart from a spreadsheet into your SGML service manual. Because most OLE objects include a Windows Metafile graphics representation of how they look, your document would show a picture of the chart looking just like it did in your spreadsheet program. And, your SGML authoring system would know how to talk with the spreadsheet OLE "server" so that when you double click on the chart in your authoring system, it launches the spreadsheet functionality and you could edit it in place (*i.e.*, your authoring system's menus are replaced with those of the spreadsheet application).

But what happens when you save the document as an SGML file and bring it to another system? SGML has no built-in way to save an OLE object (just as it has no built-in way to save a table or a graphic). There are several possibilities. Your authoring system could take the textual contents of the object and save them. It could include the binary representation of the OLE object in the SGML file. Or it could include the OLE "moniker" in your file, a pointer to the OLE object. (A moniker is a bit like a path name, but OLE objects are not the same as files; for example, a named range in a spreadsheet might be an OLE object.)

Embedding the moniker and including the binary representation of the OLE object have something in common: neither has a standard way of being expressed in SGML. So, while one SGML application may encode a moniker one way, another app will make up its own way. If OLE is an important standard — and there is no disputing that it is — then we can only hope that a body such as SGML Open will bring some standards for this to the industry as a whole, as it already has for expressing graphics.

An SGML application can also act as an OLE server, which means it can package up elements from it that are acceptable to other applications as OLE objects. At a minimum, the recipient application would be able to display the SGML text exactly as it looks in the originating app. This, however, is not a very useful capability since, for example, you might end up with text in a different font from your SGML editor embedded in the middle of your word processed document with no ability to edit it or change the formatting from within the word processor (because really all that is being displayed is the Metafile picture of the text).

As Michael Cockrill points out (see accompanying article), a more interesting intersection of the two standards might be to "wrap" an SGML object in OLE so that SGML's intelligence about documents could be used by a non-SGML application; the non-SGML app could, for example, gain the ability to do intelligent SGML querying of documents, even

though it doesn't know SGML from a hOLE in the ground. The OLE object would display the results, just as any OLE client can contain a display from an OLE server.

SGML and OLE are destined to work together. OLE enables information to be shared dynamically across applications. SGML enables document contents and structure to be expressed neutrally. With some more agreement among the vendors, they can accomplish their mutual aims — increasing the interoperability and openness of data — even more effectively.

David Weinberger

OLE, SGML, Objects and Abstraction

It is difficult to show the relation of OLE and SGML because they are at different levels of abstraction, one of the fundamental concepts of object orientation. Abstraction can be defined as separating data 'objects' from the processes (methods) that act upon the those objects. Both SGML and OLE manifests object abstraction in several ways.

For SGML, the most obvious abstraction is the separation of content from the presentation of that content. (SGML goes to an extreme in this case because the standard does not deal with how to specify a method for presentation.) Additionally, an SGML document defines a schema for how the data in that document is represented the Document Type Definition (DTD), and exposes the schema independently of the data object itself (the instance). It is through these object-oriented abstractions that SGML documents are application-independent.

The OLE standard specifies similar concepts but on a different level. While SGML focuses on the data and how that data is represented, OLE focuses on the semantics, (*i.e.* what to do with the data) and ignores the representation of the data.

The OLE standard specifies some common structures to be stored in a docfile (the standard storage mechanism for an OLE object), primarily a stream. However, it does not specify how the data in that stream is represented. The interfaces through which the data is accessed are conceptually abstracted from the data itself.

While SGML requires the user to define a DTD as a schema for a data object, OLE provides common interfaces for data access and manipulation. How these two approaches differ is fundamental to the differences and similarities between the two standards.

DTD's (combined with a declaration) specify the 'language' for a document type. They define for an SGML parser both the syntax and the grammar for the structure of a document. However, they do not provide any information about the semantics of that structure *e.g.* a DTD specifies that an elemented tagged as "h1" must proceed a series of elements tagged as "para." However, it does not specify that an "h1" should be semantically interpreted as a heading and be included in a table of contents. In true object oriented fashion, SGML abstracts the interpretation of semantics from the data itself. As with the case of formatting, the standard does not specify how to codify the semantics; this is left as an exercise to the user (hence the creation of SGML-aware applications).

Where SGML is a language for describing data objects and specifying syntax and grammar, OLE is a framework for specifying semantics. Unlike SGML, OLE does not address itself to how the data is stored (either structure or syntax), instead it concentrates on how to access, manipulate, and display data that is stored in arbitrary ways. OLE defines common interfaces that act upon data. In true object-oriented fashion, these interfaces are abstracted from the data they access and exposed to clients in a standard way.

While SGML is devoid of semantics, OLE interfaces, by their very nature, encapsulate semantics. For example, a standard OLE interface is a directive to a data object to display itself or print itself. These are directives that are meaningless to a pure data object like an SGML document.

While both SGML and OLE use the basic idea of abstraction, OLE abstracts the semantics and ignores the data representation. SGML does the opposite, it focuses on the representation of the data while ignoring the semantics. Consequently, the marriage of OLE and SGML has the potential for the best of both worlds. By storing SGML information with an OLE wrapper and exposing it through OLE interfaces, a very powerful, new data object could be created. Imagine an OLE object that exposes a query interface, either a UI or an API. A person or piece of code, interfaces with the object by specifying a query based on what the object exposes (*i.e.* the elements and attributes of the DTD). The result of the query would be extracted and displayed by the OLE object. In this way you could embed a dynamic docubase in different active document types.

Michael Cockrill

Conference Up-date

On Demand

On-demand printing is shifting to include not only just-in-time printing but also personalized, customized printing so that every user gets a document targeted to their interests and preferences. This trend was reflected at the second annual On Demand Digital Printing and Publishing Strategy Conference and Exposition, June 27-29, 1995 at the Javits Convention Center in New York City which has become the locus of activity in this field.

Over 9,100 people attended the expo, and the number of exhibitors grew 140 percent, according to the conference organizer (and publisher of this newsletter) Charles A. Pesko, Jr. of CAP Ventures.

The show floor is mainly hardware and services oriented, with large booths for the makers of the digital presses that are enabling the on-demand revolution. In obvious attendance were the print service vendors, some of whom were showing repository-based content management systems that will be very familiar in concept to readers of this newsletter.

Attending the show makes it clear — if it wasn't already — that the revolution in the way workgroups create and manage documents needs to be carefully integrated with the simultaneous revolution in the way documents are put on paper and distributed. After all, the print and distribution process is far more expensive than the document creation process. Failure to integrate these two significant phases of the document life cycle can lead to needless inefficiencies and even new hurdles to jump. (This will be the topic of an upcoming issue of this newsletter.)

CALS Update

The CALS Industry Steering Group's annual Summer Business Meeting was held in Gaithersburg on July 19-20. Since we were going to press the week before we can't tell what happened yet, but you can find out more by contacting the ISG at (202) 775-1440. Aside from the regular updates, there will certainly be a lot of discussion about MIL-STD-1840C.

The current draft of MIL-STD-1840C is dated June 12, 1995, and is prepared by and available from the Defense Information Systems Agency (DISA) Center for Standards (CFS), Code JIEO/JEBEB, 10701 Parkridge Blvd., Reston, VA 22091-4398.

One of the more interesting proposals being discussed concerns creating an 1840 media-type for exchanging information over the internet using MIME (Multipurpose Internet Mail Extensions). There is also a proposal for security enhancements to the 1840.

For more information on both of these proposals contact Ed Levinson at elevinson@accurate.com or telephone (908) 389-5550.

Industry News

Factoids

The number of homes accessing the World Wide Web rose by 50 percent in May over April. 5.1 million households in the U.S. subscribe to on-line services, up 11 percent over the previous quarter. AOL had growth of 19 percent, Prodigy 5 percent and CompuServe 0 percent. 33.2 percent of U.S. homes have a PC. (Source: NDP Group)

There are 8,000-10,0000 CD-ROM developers and 200-300 CD-ROM publishers, according to Dataquest. There were 3,000 new titles in 1993 and 8,000 in 1994. Over 60 percent of the titles are developed on the Mac, but 90 percent of them are sold for Windows. The top ten publishers have earned $50-100 million in total sales. (Source: PC Magazine)

At a financial conference, America Online said experts expect it to have five million members by the end of 1996. AOL currently has 2.3 million subscribers.

Ziffnet — being renamed to ZD Net — reports it is getting 2.5 million hits per week. (http://www.zdnet.com)

The amount of paper shipped to commercial printers has grown by 50 percent in the last seven years while magnetic storage is up by almost 1,500 percent, according to Pulp & Paper Magazine. (Source: Wired)

Adobe Acquires Frame

Adobe is buying Frame Technologies. The company claimed that the purchase gives them a solid footing in the Unix market that accounts for 70 percent of Frame's revenues. It may also be that FrameMaker 5, which heavily touted its ability to output PDF, can serve as an Adobe Acrobat publishing engine. This is an important addition for Adobe in that it gives them additional products to sell to the corporate market, which is where they are aiming Acrobat these days. Frame also gives them their own SGML technology.

Netscape Upgrades

Netscape Communications Corporation has released a beta version of its Navigator browser which includes a Windows 95 version for the first time. It is free to educational and charitable organizations and for evaluation by commercial users who are then expected to buy a retail copy for around $39. It can be FTP'ed from ftp.netscape.com.

DMA Announces Forty New Members

The Document Management Alliance (the result of the merger of Shamrock and DEN for those who may not be tracking this regularly) has recruited 40 new members. They include document management software vendors, integrators, consultants, and large corporate users. Most notable new member: Microsoft.

Lotus to Ship Beta of Next Ami Pro

Lotus Development Corporation has announced it will ship the beta of its Word Pro word processor upgrade to its current Ami Pro. The CD comes with a 45-day trial version of SmartSuite 3.1, and costs $30 which will be refunded to anyone who later buys the full version of Word Pro. Word Pro's version for Windows 95 will be available when that operating system upgrade ships. It also supports Windows 3.1 and OS/2. The software will sell for $105 when it ships for real.

Web Development Tools from W3.Com

W3.COM has announced the availability of trial versions of software packages for Web developers, including W3 WebSpin, a hyptertext page generator for database files, W3 WebForm, an application that generates custom responses to standard HTML forms, and W3 WebScan, a keyword searcher for flatfile databases. You an get further information at http://w3.com.

Visa and MasterCard Work Together for CyberCredit

Visa and MasterCard have announced they will work together to help make cyberspace safe for credit cards. At a joint teleconference, they said they'll support specifications to secure bank card transactions that will be published in September. The specifications will use RSA Data Security for encryption. Visa had been working on this problem with Microsoft and MasterCard with Netscape. The companies hope a single standard will emerge. First Virtual Holdings, a company that uses e-mail for MasterCard and Visa transactions on the Internet, said it will incorporate whatever new standards emerge.

Donnelley Manages Contents for Multiple Outputs

RR Donnelley & Sons Database Technology Services division showed the "content manager" capabilities of their PowerBase product at the On Demand show. It manages a repository of text, graphic and multimedia objects, tracks how they are used in targeted versions of different documents, and enables users to find and reuse the objects. It also keeps a "bill of materials" view of documents. A user can select the objects they want and will be output automatically to traditional print, digital print, demand fax, CD-ROM or Internet.

Donnelley's Digital Division announced Target-It which, when used with PowerBase, can build personalized communications for outputting on a digital printer; it can drop new text or graphics into a highly-designed, full-color document at each turn of the digital drum. In addition, the company announced Send-It and Order-It to make it easier to manage and use content objects a customer has stored at a Donnelley site.

Excalibur & ConQuest Merge

Excalibur Technologies and ConQuest Software have reached a definitive agreement to merge. The merger has been approved by both boards. Mike Kennedy, CEO of Excalibur will be CEO of the combined company, and Edwin Addison, President & CEO of ConQuest will be Executive Vice President at Excalibur. The ConQuest text retrieval technology adds to Excalibur's existing capability and complements their pattern recognition technology. The goal of the merger is to position Excalibur as the leader in both text and multimedia retrieval.

Prodigy Builds Home Page

Prodigy is enabling its users to create home pages by means of templates and tools that convert plain text to HTML. Initially, only text and hyperlinks will be supported, although the company plans to add graphics and sound files this summer. The six templates include "Business Card" and "Out on the Town" (a home page for describing your home town).

New Printers from Tektronix

Tektronix Inc. has introduced two new color printers, one a laser printer and the other a thermal transfer system. Both are designed for networked workgroup access and support PostScript. The Phaser 540 Plus is rated at four pages per minute in color and 14 pages per minute in black and white. It retails for US$8,995. The Phaser 240 thermal transfer color printer prints two pages per minute in color and costs US$3,695.

AT&T Builds Demand Print Service

AT&T has signed up four printer manufacturers and two software companies as partners in a networked print-on-demand service called AT&T Network Demand Printing. NDP enables customers to send documents over the network to be printed at remote locations.

It has signed agreements with printer manufacturers Agfa-Gevaert, Eastman Kodak, Indigo, and Scitex, joining with Xerox whose printers already support the network. AT&T will also offer services based on Adobe Acrobat for distributing formatted documents across systems. Quark will put its multimedia viewer on the AT&T service as well. The service is expected to be available in the first quarter of 1996, initially for black and white documents. Charges will be based on usage.

Bitstream Creates OpenDoc Fonts and Signs Novell

Bitstream has joined CI Labs, the OpenDoc association, and will provide fonts to support the OpenDoc cross-platform architecture. It is one of 25 new members of the consortium. The Bitstream TrueDoc fonts are in alpha test.

Separately, Bitstream announced that Novell will be incorporating its TrueDoc technology into Envoy, Novell's Acrobat competitor.

Knowledge Access Shows CD-ROM Publishing and Storage

Knowledge Access has demonstrated CD-ROM solutions for publishing and document storage. OmniSearch Disk Publisher is an index and retrieval system for Windows. CD-Rchiver is a turnkey PC-based document archiving system.

Standard Register Resells Saros

Standard Register and Saros Corporation have announced a strategic partnership under the terms of which the former will resell Saros Mezannine as the engine of its DocuStream "document automation" system. DocuStream provides services and technology to manage and coordinate paper and electronic systems operating in parallel, including forms. No release date was announced.

SoftQuad Now Shipping HoTMetaL Pro 2.0

SoftQuad Inc. has just announced that it is now shipping HoTMetaL PRO 2.0. The company's HTML editor for MS-Windows, with other platforms soon to follow.

Buzzer of the Month

In an unusual move, this issue's award for best use of buzzwords in a press release goes to I & I Specialities for the Best Trivialization of a Dream:

"JACKSONVILLE, FLORIDA, U.S.A., 1995 JUN 29 (NB) — Once upon a time futurists believed the paperless office would be commonplace by the year 2000. Today even people who produce the imaging systems that would make that dream a reality admit it may never happen, but a Florida company has introduced a way to let the paperless and paper-centric worlds co-exist.

"I & I Specialties Inc., has introduced Stock-A-Disk, a diskette-size envelope with an adhesive back and resealable flap that lets computer users store 3.5-inch floppy disks in the same file folder with the printout that the boss demands."

People News

Ann Palermo has joined PC Docs as Vice President of Marketing. Ann was with IDC.

Rich Kennewick and **Rob Richardson** have left Interleaf for Saros.

Edward Sanderson has joined Oracle as Senior Vice President of Oracle Services and will manage the commercial consulting activity in North America.

Stephen A. Schoffstall is now Vice President of sales at PSINet.

Dr. Donald A. Norman has been named Vice President of Apple Computer Inc.'s Advanced Technology Group, reporting to **David Nagel**, Senior Vice President, Worldwide Research and Development

Pat Condo was recently promoted to President at Excalibur, reporting to CEO **Mike Kennedy**.

Karl M. Freund has been named to the newly created position of Vice President of Marketing at Cray Research Inc.

Action Technologies has appointed **Terry R. McGowan**, President and CEO.

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 topics covered in our report.

CAP Document Management & Electronic Delivery Seminars. Fall dates TBA, London, UK. These two day seminars are conducted by Gilbane Report staff and are managed by Technology Appraisals. Call +44 81 893 3986 or (617) 547-2929, Fax +44 81 744 1149 or (617) 547-8811.

CALS Korea '95. September 18-20, Seoul, Korea. Expo and conference covering CALS activity. Call (510) 522-5077, Fax (540) 522-1228.

Seybold Seminars '95. September 26-29, San Francisco, CA. The annual conference where the publishing technology elite gather. Focus is on pre-press, color, newspaper, and magazine applications with some corporate application coverage. Call (415) 578-6990, Fax (415) 525-0183.

CALS Europe '95. October 4-6, Hamburg, Germany. Annual Pan-European expo and conference on CALS technology and applications. Call (703) 578-0301 or +49 30 882 6656, Fax (703) 578-3386 or +49 30 881 5040.

Document Management and Imaging Expo Tutorials. October 23, New York, New York. Introduction to document management and imaging as well as workflow and business process redesign. Call (617) 837-7200, Fax (617) 837-8856.

Document Management and Imaging Expo. October 24-26, New York, New York. The premier document imaging event on the east coast. Call (207) 236-8524, Fax (207) 236-6452.

CALS Expo '95. October 23-26, Long Beach, CA. The annual expo and conference covering CALS activity in the U.S. and internationally. Heavy defense industry emphasis. Call (202) 775-1440, Fax (202) 775-1309.

Xplor '95. November 5-10, Minneapolis, MN. The large 16th annual global gathering of electronic printer users and vendors. Call (800) 669-7567, Fax (310) 373-3633.

Documation Canada Conference. November 13-15, Quebec City, Quebec. Documation for the Canadian market. Organized by InterDoc with CAP Ventures and Graphic Communications Association. Call (514) 288-7501, Fax (514) 288-7596.

SGML '95. December 4-7, Boston, MA. Sponsored by the Graphic Communications Association. Call (703) 519-8160 for more information.

Documation France Conference & Workshops. December 11-14, Paris, France. Sponsored by Techno-Forum SARL. These two day seminars are moderated by Frank Gilbane, Yves Stern, and Guy Fermon and cover compound document management and electronic delivery technology and trends. Call +33 1 43 48 57 92 or (617) 547-2929, Fax +33 1 43 48 55 43 or (617) 547-8811.

Documation '96. March 11-14, Long Beach, CA. The big annual conference and exposition covering document management systems, technologies issues and trends. The best single source event for learning about Document Management. Co-sponsored by CAP Ventures and Graphics Communications Association. Call (617) 837-7200 or (703) 519-8160, Fax (617) 837-8856 or (703) 548-2867.

Topics Covered In Previous Issues

Vol. 1, No. 1.
What The Report Will Cover & Why — An Introduction To "Open Document Systems", And A Description Of The Report's Objectives.
Imaging, Document & Information Management Systems — What's The Difference, And How Do You Know What You Need?

Vol. 1, No. 2.
SGML Open — Why SGML And Why A Consortium?
Document Query Languages — Why Is It So Hard To Ask A Simple Question?

Vol. 1, No. 3.
Document Management & Databases — What's The Relationship?

Vol. 1, No. 4.
Electronic Delivery — What Are The Implementation Issues For Corporate Applications?

Vol. 1, No. 5.
Multimedia Rights & Wrongs — What IS Managers Should Know About Copyrights In The Age Of Multimedia.

Vol. 1, No. 6.
Document-Centered Interfaces & Object-Oriented Programming — How Will They Affect You?

Vol. 2, No. 1.
State Of Wisconsin Legislature TEXT2000 — Reengineering For Document Management.

Vol. 2, No. 2.
Document Management Industry Update — Documation '94 & Other Spring Industry Events.

Vol. 2, No. 3.
Document Formatting Interchange — Why Don't We Have A Solution?

Vol. 2, No. 4.
Corporate Publishing On The Internet — Is It Realistic Yet?

Vol. 2, No. 5.
CGM: SGML For Graphics? — A Structured Vendor Neutral Interchange Format for Graphics.

Vol. 2, No. 6.
Interoperability Standards — What Are They and How Do They Relate?

Vol. 3, No. 1.
Object-Oriented Document Database Systems — What Are the Benefits?

Topics Covered In Upcoming Issues

Vol. 3, No. 3.
Document Management and ISO 9000

Vol. 3, No. 4.
The New Electronic Document Distribution Landscape — Life After the Web

Order Form

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

U.S.A.: $225 Canada: call InterDoc @ (514) 288-7501 Foreign: $242

Additional copies and site licenses are available at reduced rates. Call for information.

Please send me additional information on:
o Documation Events o Market Research
o Strategic Consulting Services
o Compound Document Management Consulting Service
o Document Management & Electronic Delivery 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: Please call for details.

Name _____ Title _____

Company _____ Department _____

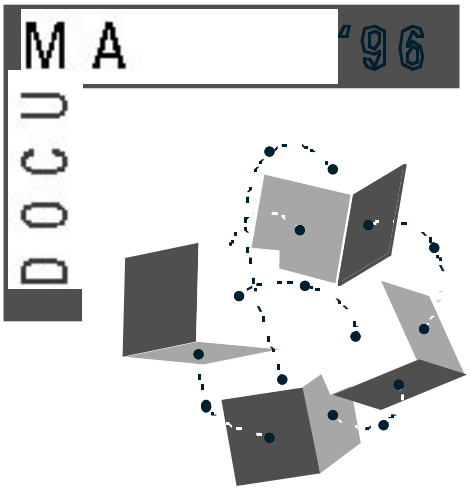
Address _____

City _____ State _____ Zip _____ Country _____

Telephone _____ Fax _____ E-mail _____

Mail or fax this form to:
CAP Ventures, One Snow Road, Marshfield, MA 02050
Fax: (617) 837-8856 • To order by phone call: (617) 837-7200

MARK YOUR CALENDAR!



DOCUMATION '96 CONFERENCE + EXPOSITION

March 11-14, 1996 — Long Beach, CA

PLAN NOW FOR THE DOCUMENT MANAGEMENT INDUSTRY'S *BIG* EVENT OF THE YEAR!

Call, Fax, Or E-mail Us To Find Out More About Events & Companies Mentioned In This Issue

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, CAP Ventures, One Snow Road, Marshfield, MA 02050 or frank_gilbane@capv.com

ISSN 1067-8719