TEXT TOOLS: BEYOND SEARCH & RETRIEVAL

The market for intelligent manipulation of text is about to reach critical mass. It has been growing slowly for many years now, the province of academics, publishing firms and slow-to-materialize giant documentation projects for aerospace manufacturers and Defense Department contractors.

In this issue we briefly note some factors that will spur the growth of this market. Then we discuss the underlying technology -- treating text as objects and the role of the SGML markup language -- and a number of interesting tools and companies that will benefit from and foster the market's growth.

Text tools are about to address a much broader market than the high-end specialist customers of yore. Formerly high-priced, special-purpose, proprietary systems, they are now becoming robust standards-oriented tools that run on pcs and UNIX and can interoperate with generally available databases, word-processors, and the standard commercial dp environment. "We're selling to MIS people now; we used to just sell to publications departments and contractors," says Haviland Wright, founder of Avalanche Development, one of the key technology players (page 14).

Another major spur to activity is the potential opening of the information services market to the regional Bell operating companies (RBOCs). Last week Judge Greene, the man who helped break up AT&T, tentatively cleared the way for AT&T's progeny to offer information services. Up to now they have been restricted to distributing such services for others; generally, they can transmit information only so long as they don't provide the content, or select, alter or otherwise add value to it.

Although final resolution of the issue is still at least a year away, the telephone companies are likely to win the long-awaited right to create and sell, rather than just distribute, information services over their facilities. Chances are that within a year most phone companies will be scratching around to come up with services to offer. Should they revive videotext? Run their own hot lines? Offer moderated teleconferences? Get into electronic bulletin-boards to compete with CompuServe, or whatever-it-is to compete with Prodigy?

Even if the phone companies don't win their way, the lobbying over this issue will raise public awareness of the potential for online

IN MEMORIAM BOB BERLAND

INSIDE

BEYOND SEARCH & RETRIEVAL 1
A market emerges.

DOCUSOURCE 4
A model for the Bells.

SGML PRIMER 7
Text: objects of desire.

TEXT TOOLS 14
Text tags: Avalanche's FastTAG.


Objects: Interleaf's Active Documents, Pages, Accurate Information & Ontos.

PHONES/RESOURCES/CALENDAR 25-27
Names, phones, books, dates.
services involving text and multimedia as well as voice, with both camps promising all kinds of services that they will offer if they aren't kept out of the market by predatory RBOC actions or by exclusionary government rules. Either way, someone will be offering such services pretty soon. Whether it's the Baby Bells themselves or their competitors building and using such software (along with outside tools), networked information services promise a huge market for the intelligent manipulation of text. (See also Release 1.0, 4-91, about standards for online search and retrieval as opposed to content manipulation.) Overall, the RBOCs represent a pool of money and a distribution system -- and a huge market for software, not just a competitive threat.

The line-up

We begin this issue with a description of Bell Atlantic's DocuSource, an early foray by an RBOC into a large-scale, networked text management system. Next we look at the underlyingly technologies -- SGML and text objects -- and finally we consider a number of other text tools. Some of them use SGML; some don't. In fact, one of the most widely used tools is Avalanche's FastTAG, which has a solid role as a conversion and tagging tool precisely because SGML and other conventions aren't yet in wide use. A wide range of hypertext systems handles text for distribution to users: DynaText from Electronic Book Technologies is a hypertext compiler, generating online hypertext automatically from SGML-tagged texts. IBM's Book Manager and Teleprint IDDS also compile tagged files for read-only document delivery. Guide from OWL International is a more flexible but less automated system, more like a wordprocessor, that handles a variety of inputs. SmarText from Lotus/Samna automates creation of logical links, not just creation of electronic hypertext from pre-specified links. Folio Views and RDI's IZE are other interactive hypertext tools.

On the text-as-object front, Interleaf's Active Documents illustrates the potential of treating text as objects. The forthcoming product line from Pages (unfortunately not yet available, or this newsletter would start looking elegant) is a neat example of the manipulative power you can get from dealing with objects instead of raw text and images. Finally, there's the Accurate Information Systems project using SGML and the Ontos object-oriented database, which shows how OODBs can be used for text. (There are also a number of SGML-oriented word-processor/parsers, from SoftQuad, Datalogics, Exoterica and Exoterica OEM Arbortext, not covered here.)

In part the simultaneous emergence of the RBOCs, the growth of other online services vendors, and the emergence of standards and technology for text manipulation are just a coincidence of timing. But of course they all drive each other. (Twenty years ago the phone companies might have gone into what was known as time-sharing -- database management and accounting services that everyone is now doing inhouse. Instead, data-oriented services, notably financial information and transactions which grew up over the past two decades, are now offered over phone lines by third parties.)

Release 1.0 31 July 1991
Text and multimedia are uniquely suited to online distribution, because so much of the information is worth disseminating rather than keeping inhouse (unlike accounting and personnel records). Product documentation, electronic communications and plain old text databases are mostly meant to be shared with other people -- and are much more complex to handle than plain old numerical data. (The transfer of numbers comes under EDI, or electronic data interchange of purchase orders and the like, also mostly run by third parties.) One more reason for the phone companies' interest in text is their widespread use of documentation for products and procedures, and their experience with publishing and advertising through phone books.

The flaws in current electronic information offerings include the difficulty of reading plain text, the multitude of interfaces and formats once you get beyond plain text, and the difficulty of filtering the wheat from the chaff. Applications of SGML and other intelligent text tools promise a solution.

Military movements

The other driving force is the Defense Department's CALS (for Computer-aided Acquisition and Logistic Support) initiative. Among other things, CALS directs all vendors to provide proposals, documentation, parts lists, training and repair manuals and all other materials in standard, electronically revisable form. There are several major government information systems projects up for bid, each ultimately worth hundreds of millions of dollars to the selected vendors and subcontractors over the next 10 to 15 years:

- **Army CALS** -- a 15-year contract to convert most of the service's engineering documentation into standard, revisable, electronic form. Xerox and Computer Sciences Corp. are vying to be prime contractor; the decision should be made by this fall. Many of the companies listed here are subcontractors to either CSC or Xerox or both of them.

- **Air Force RFP for 902-S** -- a program for the Air Force Information Publishing Service, akin to Army CALS, but not so far along.

- **DMRD 998** -- Defense Management Review Directive 998, which directs the Navy to consolidate all Defense Department printing (as opposed to electronic distribution). This will start with an RFP for SGML services and research at the Navy's David Taylor Research Labs in Bethesda.

- **JUSTIS** -- Joint Uniform Service Technical Information System, a tri-service project to unify documentation for items used across the services, avoiding redundancy and improving consistency. A draft RFP is expected in August, but the project may be merged in with Army CALS.

- **DMAC II** -- Departmental Microcomputer Acquisition II, a long-disputed procurement for the Treasury department (including the IRS), finally resolved this month with an award to reseller Sysorex. The total could be worth $400 million over four years, possibly as much $16 million of it to SoftQuad, a Toronto vendor of Author/Editor, an SGML text-processor (see Release 1.0, 1-89). (The other equipment is mostly widely used products such as pcs, Macs and other packaged software.) The contract was the first government procurement to specify SGML, and is notable for not being for the military.

*Release 1.0 31 July 1991*
The standards

The standards include SGML (Standard Generalized Markup Language), for describing text objects and document structures; DSSSL (Document Style Semantics and Specification Language), a proposed standard for formatting and layout commands, which takes up where SGML leaves off in preparing documents for output; and FOSIs (for Formatting Output Specification Instances). For describing formatted documents (rather than revisable content) there are CGM (Computer Graphics Metafile), and SPDL (Standard Page Description Language), close to PostScript but page- rather than file-oriented.

THE RBOCS -- AND DOCUSOURCE IN PARTICULAR

Any service the RBOCs offer will ultimately make sense only on a grand scale, but it still shouldn't cost too much so that it won't need to be subsidized by local rates, which are still regulated. That argues for software-enhanced services, rather than ones that require people to be standing by. Software can be scaled up easily (as long as it works!) to handle arbitrary levels of use, and the economies of scale are appealing. In fact, this is what makes telephone companies so attractive as candidates to offer these services -- and so frightening to those they may compete with: newspapers, cable tv services, existing electronic information vendors and software companies.

We're used to systems integration for database-oriented applications, where a large number of applications share data. However, until now most text-oriented systems were single-purpose. Was that because people couldn't think of what to do with text, or because it was stored in special-purpose forms? DocuSource is a harbinger of the standards-driven integration of a collection of text tools into a single distributed system that can manage text from words on the writer's screen to words on the reader's screen.

The most promising case in point is DocuSource, a project-turned-product of Bell Atlantic, developed under its Champion "intrapreneurial" program (which also brought us Thinx, an intelligent graphics program). DocuSource is an inhouse project built mostly with tools, software and even development efforts purchased from outside. (And Bell Atlantic will also use its development partner, OWL International, as a reseller of DocuSource.)

DocuSource exemplifies the next generation of text-oriented tools that will exploit the potential of wide-area networks -- the role intended for Xanadu, the long-awaited hypertext server from Autodesk that is now promised for shipment next year (see Release 1.0, 7-89). Ultimately, there will be ample demand for multimedia as well -- fostered by tools such as MacroMind Director and hypertext-turned hypermedia such as OWL's Guide, and standards such as SGML for text and HyTime (an SGML application) for multimedia.

Release 1.0 31 July 1991
Platform for electronic publishing

DocuSource began as the revival of a mid-Eighties project for online documentation which was abandoned by Bell Atlantic because the necessary technology was unavailable -- or at least too off-beat to introduce into Bell Atlantic. In 1988, however, Jeff Beegle, an engineer who had worked on the original project, proposed to revive the project as a commercial offering under Bell Atlantic's Champion program. After a concept and market study, he got funding late in 1989 to proceed with the project full-scale.

DocuSource is now ready to launch commercially, in the form of Release 1.3. It is already in use within Bell Atlantic's human resources department for a database of about 5000 job descriptions, including salary levels (by category, not for individuals!), requirements and so forth. A second project -- basically an online procedures manual for HR functions such as screening applicants, hiring, firing, transferring and promoting employees -- is planned. A third will serve the medical department, with instructions for handling ailing employees, policies on medical care and so forth.

DocuSource is also being evaluated at a number of outside commercial sites that aren't ready to be identified yet. They include not just inhouse users but a publisher who sees DocuSource as a possible way to prepare and maintain information for electronic distribution. As it is now, Bell Atlantic can offer the software or processing services, but it can't deliver the results online. If the RBOCs get their way, DocuSource could also be a hypermedia delivery service run by Bell Atlantic.

What is DocuSource? Basically, it's a full-scale system for preparing, editing and delivering text and images online. It takes text and image files, processes them to produce hypertext complete with cross-reference links, full-text search, images and (soon) laser-disc video. In structure, it's a collection of tools tied together by Beegle's team of inhouse developers. The basic hypertext engine is OWL's Guide, enhanced by OWL (page 17) under contract. (BA owns the software thus developed, but OWL will also resell DocuSource under a license agreement and pay BA royalties.) The engine works with text objects and links predefined by document creators or recognized by Avalanche's FastTAG (page 14).

The system uses OWL's Guide Reader to deliver the results to users through a Windows interface. Users can view the hypertext, search for specific sections or topics via the table of contents or an index, or jump from one place to another following hypertext links.

...and the financial infrastructure

And, unlike any other fielded system we have seen, DocuSource includes facilities for access control and fine-grained management of payment to individual copyright holders (although American Information Exchange, with bid-and-ask pricing, is on the way; see Release 1.0, 7-90). These are key capabilities for hypertext as a medium for publishing information from a variety of sources outside the confines of a single licensed user organization. Using DocuSource, vendors of information can specify the charges for each type or duration of activity -- opening (viewing), copying or printing. (Once something is copied, the vendor is dependent on the user's honesty or another monitoring system.) Please turn ahead to page 6.

Release 1.0
31 July 1991
Above, left, is a screen that lets the builder-user compose messages outlining the various usage and charging options for individual pieces of copyrighted material. At right, the builder can specify which user actions to log. Below, left, the user gets a warning screen before he spends any money. Below, right, DocuSource offers a clever interface for text retrieval: Rather than just lead you to each occurrence of a word sequentially, it shows you the number of occurrences of the words "docusource" and "hypertext" (shown in the band over the bottom window) in each section of the document listed in a table of contents (top window). You can then go to the section you consider most relevant, based on the section's title and the number of word hits.

Release 1.0

31 July 1991
Integrated tools for integrating texts

Where DocuSource and the other tools become interesting is not for the creation of hypertext out of a single document (see SmarText and DynaText below for that) but for their ability to meld texts in real time from a variety of sources and suppliers into a consistent pool of texts that can be retrieved by the viewer as a single, coherent whole from which to pull subsets and assemble documents -- assuming that Bell Atlantic (or someone else) can offer it as a service over the phone lines (its own and other carriers'). Note that BA would not necessarily be a supplier of the base information, but it would be adding value to it and integrating it through the DocuSource capabilities; that is what is still prohibited by the Modified Final Judgment, the AT&T consent decree.

Meanwhile, there's no need for each text supplier to worry about layout and formatting. Sections from different original suppliers will look the same -- following the formatting and layout conventions of the particular DocuSource system builder. (The conversion process takes place beforehand, not in real time, although the text can be assembled in real time as the user makes choices and follows links across different documents.) Pricing starts at under $10,000 for a minimal start-up authoring system, but any working production/delivery system would cost much more.

SGML PRIMER: WHAT EVERY SOFTWARE COMPANY SHOULD KNOW ABOUT SGML

Before we go ahead, a little bit about the underlying technology and its best-known standard, SGML.

SGML stands for Standard Generalized Markup Language. It is actually a syntax for building programs rather than a single language; there are SGML implementations rather than "an SGML." You use SGML to describe and define text elements and to create a Document Type Definition (DTD), a set of terms for text elements and a program to define how the elements are organized.

The DTDs provide a data structure for the SGML objects -- document frameworks, if you will. The relations between the particular classes of elements are defined, although the number and perhaps the presence of the instances varies from document to document. For example, a DTD could specify that a chapter begins with a heading and may contain any number of paragraphs, pictures and associated captions, two levels of subheads. Other tagged text elements might be index terms and customer names (which could be used to retrieve customer addresses or order amounts).

Some common SGML DTD standards are the DoD's Mil-M-28001, ATA-100 for the aerospace community (see Release 1.0, 4-91), and the American Association of Publishers' Electronic Manuscript spec. DTDs can manage common, defined sets of text-item relationships, sort of templates for certain kinds of documents, just as letters have addresses, salutations, post-scripts, enclosures and so on; books have titles, chapters, blurbs, perhaps indexes and tables of contents; legal documents have carefully defined sections, footnotes, and possibly claims, counterclaims, exhibits, related depositions, and so forth.

Release 1.0
31 July 1991
Documentation may also have its own model, with links between references to parts, sections for different versions of the same part or parts that don't exist on all versions of a product. A catalogue has references to parts and prices that keep changing.

These DTDs are *models* of elements and structures, not programming languages or applications. Building a DTD is akin to using SQL to describe data elements and the structures of tables in a database. However, in SQL the only possible data elements are records, fields and tables, whereas SGML allows you to define arbitrary data elements. Also, with SGML there's the notion of context: A table is a table is a table, whereas a paragraph within a caption may be treated differently from a paragraph in the body text.

These elements, once defined, are typically stored in a document, and are identified within the document, which serves as the data store (at least until object-oriented databases take over). The structure of the document must comply with the DTD, or there will be trouble later when an application tries to parse and process it.

A DTD is a fine data storage structure if the texts are going to be used only in a few documents of basically similar type; when you start looking for more complex reuse, an object-oriented database is better. (See Accurate Information, page 23.) You can also use a relational database or SFQL (see Release 1.0, 4-91), but an OODB provides a better match of content and structure. On the other hand, a relational database is a fine place to store data that may be queried by scripts in text objects, such as prices from a catalogue.

The value of text objects is not so much reusable code as reusable *data*. You can reuse defined pieces of text in different documents, taking chunks from your business plan to reuse later, say, in a marketing brochure or an annual report. The documentation for a component can appear as part of the manuals for all the pieces of equipment that use the component. And you can update all the uses from a single place. From the other side, text chunks from different sources can be collected and integrated into a single document. Text objects that need updating can be updated easily without losing their identity in the documents they are part of. And changes in formatting can be applied globally to precisely the right text components; you may want to change italics to boldface in Formal Titles, but not where they were used for emphasis.

OODBs and text

These text elements are passive objects, which can be manipulated by any application that understands them and the DTD they conform to -- whether printing commands (straight translations from an SGML markup) or other more complex procedures. But note that SGML/DTD tags don't make the text elements into objects. The tags simply note their presence in a document so that they can be treated as defined data elements (passive objects or scripts) by a procedural program. Or they can be instantiated as true objects by a full-fledged object-oriented system, which provides active methods for them to implement. To the extent that the marked text elements are
objects in the sense of inheritance, methods and encapsulation, they are defined outside SGML. (In other words, the document contains the instances of text objects, along with tags that identify them. Those tags are listed in a DTD that specifies the document's structure. Those same tags are the names of classes in an object-oriented hierarchy that contains the classes and their behavior, or methods. You could also store the objects in an object-oriented database, in which case a document would be just one representation of a subset of the class instances. Likewise, that document would be just one instantiation of a particular DTD.)

Documents and spreadsheets

To clarify by analogy: Consider a text base as a database. The individual documents are the equivalent of spreadsheets using data downloaded from the database. You get a lot faster data-manipulation performance from a spreadsheet than a database, and a more intuitive feel. But it's important to have that database at the back end to maintain data to support multiple users, as well as a variety of spreadsheet models with the same data but different assumptions, or certainly different subsets and organizations of the data. Moreover, that database can also provide data for graphs, mail-merge, queries and reports... So, is it important to support SGML? Is it important to support SQL? Is it important to be SGML-based? Is it important to be SQL-based? Just as the world moved to SQL, we believe, it will also move to SGML -- since any standard wins against a vacuum. (Call it the Mario Cuomo of text-processing?)

With SQL, the standard data structure is a table, with columns and rows (or tuples). But with SGML, the structure is part of the information that's unique to each case. (That doesn't mean that your data has to retain the same structure when you make a view, but those relationships are part of the data, rather than a function of the data's values.) Thus, a headline is linked to the text that follows it, not by a value, but because that's a relationship embodied in the document and made explicit by the DTD.

A short history of markup

Markup started as a paper publishing issue, so that printers could know how to handle the various text elements. In fact, the predecessor to SGML, Generalized Markup Language, was a formatting language developed at IBM and encouraged by the IRS, which wanted to use it to make the huge volumes of texts it was creating portable across platforms. At that point, anytime you used the markup (formatting) commands of a particular word-processor, you were tying yourself to that word-processor and the machine it ran on. You were also tying yourself to 14-point type for the headlines, hanging indents for the bulleted sections, and so forth.

Traditionally, one thinks of markup as specific instructions, such as "10pt bf" (for 10-point boldface) or "skip three lines, indent 5 spaces." This is known as procedural markup. The problem is that it is not smart or abstract. It doesn't really define the text; it just says what a formatting

---

1 For a complete, lucid discussion of this issue, see the excellent article on "Markup systems..." listed in our resources section. If this newsletter were hypertext, we'd make sure to link to it with a must-read link.

Release 1.0 31 July 1991
program should do at a certain point. That's fine if all you want to do is format the text once. But if you want to reuse it in several different documents, edit it with a new word-processor, change your style conventions, or perhaps do something more complex with a document's contents, procedural markup doesn't cut it.

Descriptive markup

Descriptive markup identifies the text elements without immediately specifying what to do with them. For example, descriptive markup says only, "This is a paragraph." With descriptive markup, a formatting program can still automatically format the text just as with procedural markup. But the same descriptive markup and source files can also be used with a variety of systems; it retains its identity regardless of how printing programs, formatting commands and output environments change.

With descriptive markup such as SGML, for example, you could display the same document with different formatting and page sizes and graphics, on a Sun workstation, a cheap terminal hooked up to a mainframe over a modem or to a pc, or even a PenPoint machine (once it acquires this capability from some canny vendor). The user experience would vary from system to system, to be sure, but the content would be the same both for the user and for the developer, who would need to provide only one source file for all environments. Conversely, a variety of different applications, on the same or different computers, could transform the mark-up into very different user-specified formatting commands (style sheets applied to objects instead of to locations within the text); each implementation would look very different.

Further than formatting

But most significantly, descriptive markup allows for huge flexibility and extensibility. This power becomes more important as we move to electronic distribution of texts. With descriptive markup it's possible to identify not just text objects for formatting, but terms for inclusion in an index, headlines for inclusion in a table of contents, footnotes for either concurrent or end-of-book placement, and cross-references for resolution into page numbers. A good text-processing program can resolve these to the proper page number or chapter title or diagram number, even if the user has changed the precise words within the object in the meantime.

Thus, an index is actually an alphabetical list of cross-reference links pointing back to the location of the tagged index words in the text. A table of contents is a sequential list of the titles and headings down to whatever level the user specifies, usually a separate file in documents destined for print output. (By contrast, although the difference isn't apparent to the user, in a hypertext document, the table of contents is usually the unexpanded, top-level form of a document, like the top few levels of an outline. Expansion links, explained below, bring the body of the document into view; the body, of course, contains many links of its own.)

More complex procedures are also possible. Tagged elements can contain executable commands; for example: "Here's a link to another marked item elsewhere in the text. Get its value, insert it into the text here." They can outside the current text, such as "Go find the current value of OVERDUE_BILLS in a database or cell 59F in a EXPENSE.WKS, and insert here." Or the
script within a tag could even instruct the system to load another application, perform an action, and insert the result, or use the result to determine which paragraph to display next. You could say things like "Omit this section if context = 1," and set a context for a document that could be determined at runtime. Anything is allowed, because it's only a language.

To catch a tiger, put salt on its tail...

To do interesting things to text you must first be able to identify it in ways meaningful to a computer. A page of text may be meaningful to a reader, a layout expert, a proofreader or a customer service manager, but to a computer it's just a string of ASCII. Even a formatted page is still just a sequence of text with embedded formatting commands. Those commands may be complex and elegant, such as PostScript, but they have to do with the graphical representation of elements rather than the intrinsic components of a document.

But now, a "document" has become a more interesting concept. Until recently there was just the notion of the document as something linear. But in fact it is just a display -- on screen or on paper -- of a subset of a potentially larger body of matter. Suddenly the document became modularized into components. You could in fact store them in a database, not in a linear sequence at all, with numbers or other values to indicate the sequence they come in.

It may be a selection of chapters from a book, the relevant parts of a manual, an insurance policy or sales proposal, or even a French or Russian version of a superdocument that could be rendered in any of many languages. It may be information that could be expanded into a news story or contracted into an earnings table, organized on a time line or classified by country first. A database expert would immediately recognize this as a view -- a temporary construct created by selecting and organizing a subset of items form a data table -- or perhaps by joining the contents of one or more tables.

To do that, you need a data (text) description language, a data (text) manipulation language or application, and some text...

The analogies of text to data are illuminating, but not exact. But they hold the promise of a proliferation of tools that can handle text as powerfully as data, allowing us to apply the same efficiencies to dealing with text as we have to databases. Of course, text is more complex, meanings are more nuanced and so forth, but these challenges merely raise the value of solutions. (See also Release 1.0, 3-90.)

Living links

It gets more interesting if you're not just printing a document but presenting it live to a reader -- or viewer. This, of course, is the foundation of
hypertext. As hypertext has come into vogue (in some circles anyway), SGML has become a possible way to define items for linking, and to represent the links themselves. Although most of the hypertext systems use proprietary coding and tools, their customers are beginning to appreciate the flexibility and portability SGML offers.

These live links implement the fundamental power of hypertext. Unlike text cross-references, which tell the user where to go for further information, they do the work for the user -- as described in this unnumbered list\(^2\) (precise terms used by each vendor vary):

- **expansion link**, which directs the program to insert the linked text in-line.
- **reference (go-to) link**, which takes the user to the document, location or other section referenced.
- **note link**, which instructs the document to display the object referenced in a box or window while leaving the original display on the screen.
- **action link**, which can call on other applications to perform arbitrary tasks, including loading updated data into the text, checking on a context so as to change the formatting or display or not display a marked section. A special case is multimedia links, which may cause the playing of audio or video sequences. (HyTime is a proposed standard that includes multimedia links and allows for the incorporation and synchronization of timed information -- sequenced images, video or sound, basically -- into a document. HyTime, an SGML application/extension, adds the fourth dimension.)

And beyond that into applications

Links (and other text objects) can also be typed arbitrarily by users, or clever algorithms, for other processing by applications. For example, there can be supporting and dissenting annotations, or comments classified by author, a common feature in many editing/annotation systems. Paragraphs can be classified by topic, determined by word statistics (as in SmarText) or more clever algorithms (IZE and Verity Topic). For storage in an OODB, for example, you might want tags that say this is a paragraph about a widget-assembly screwdriver, or a customer name.

SGML is really a language for defining objects (though not their behavior), just as a data-definition language defines data. The difference is the underlying data structure -- and just about everything else. While data instances are defined by their values -- records with fields matching certain values, say -- textual data and objects are frequently defined by where they

\---

2 Note the formatting, which XyWrite represents as "<<ip 3,5>>." It's up to a reader or a conversion tool such as FastTAG to figure out that it's an unnumbered list. And unfortunately, at Release 1.0 we have neither FastTAG nor a formatting tool to tell a program to use underlined italics for each list element up to the first comma; instead, we have to put in the formatting commands by hand for each item.
are (their relationships to other objects): the fourth paragraph in the second section, the fifth footnote, the caption under the chart numbered 5-19. The objects, moreover, don’t change their identity when they change their text contents: A rose by any other name would still be the same object.

Text applications include not just layout, but almost anything to do with management of information. The truth is, once text is defined as objects, you can do anything with it that you can express explicitly. As we noted in our issue on scripting (5-91), defined objects let you benefit from the powers of scripting (or programming).

SGML as a standard

SGML identifies text objects in a global way so that they can be used across applications. Its capabilities aren’t unique, of course; Interleaf’s Active Documents, for example, has them and more, but in a proprietary way. With SGML you’re not dependent on a particular environment, but can use any one that supports SGML (although of course you need whatever application facilities your active objects require to act). That is, the value of SGML in particular depends on its status as a standard. Whatever its flaws, that’s a fact of life, and there’s no real contender out there as yet that could unseat it.

What makes SGML so valuable for handling information is that it was specifically designed both to sit inside text, and to surround and define objects within a text. And text is information. Yet the power of SGML also depends on its flexibility. You can define objects, and then define transformations or tasks to perform on those objects (or use existing applications). SGML can be stretched way beyond the original concept. As Accurate Information Systems’ Rita Knox says: "The power of SGML isn’t in the language itself; it’s in what’s doing the parsing and the executing."

---------

Juan & Alice do hypertext R & D

Alice:  Do you think anyone has ever made links like this before?
Juan:  Not to worry. A few weeks of development and testing can often save an afternoon in the library.

-- source unknown, courtesy of Bruce Webster, Pages

---------

Release 1.0 31 July 1991
AVALANCHE DEVELOPMENT'S FASTTAG (It knows an object when it sees one)

Most authors don't bother to tag their text, or they use a specific formatting tool that's different from the one required by most other delivery systems. This presents difficulties (which wouldn't exist if the world were perfect) for anyone sending documents from creation to mark-up to formatting and print or onscreen delivery through all but the most integrated text-handling systems -- and a huge opportunity for Avalanche Development.

Avalanche sells FastTAG, a customizable object-recognition tool OEMed by many of the players here. FastTAG recognizes text objects and marks them up, either with SGML or with whatever customized tags a customer requires. The company now has 18 people and earned revenues of about $1 million last year, mostly from OEM sales to customers such as Xerox Information Systems, Bell Atlantic (DocuSource), IBM and DEC.

FastTAG works on scanned-in, OCRred text, which has both content and visible form, on plain ASCII, and on the output of a variety of word-processors and high-end systems including Interleaf with various embedded formatting commands. It uses a configuration file customized for each kind of input source file, Inspec (for Input Specification), to generate a visual representation of the text. It parses that representation for objects, although it saves the source-file encodings (such as footnote commands or table markers) to help generate the target-file encodings later on.

Text-object recognition is a typical AI task. First the tool picks out sections of text and graphics -- typically blocks, but not always. Examples are headlines, paragraphs and page numbers. FastTAG also handles tougher items such as captions, tables, lists (with bullets or numbers), inset quotes, legal or bibliographical citations. (Footnotes are easy to pick out in ASCII or formatted text, but tough in scanned text because scanners tend to lose horizontal separators and the footnotes just look like paragraphs.)

This goes well beyond two carriage returns equals a paragraph (or a headline, if there's a font change). The invariable phrase at the top of each page must be a title or chapter heading -- but which? In essence, FastTAG tries to reverse-engineer all the information a text contains beyond the characters themselves. Thus it uses any clues a person might use -- analyzing sequences of numbers to determine if they are content or a diagram's ID within a text, as in "(See Figure 5-9a)," which could be resolved into a cross-reference. It's usually easy to recognize a table, but what's the heading and what's the body information -- especially if a table runs a few pages? And what about "cont. on page 126," which appears on page 119, just before the table?

Once FastTAG has recognized the objects, it uses a second configuration file, Louise,3 customized for the output file required -- whether it's for a composition system, a hypertext tool, a word-processor style sheet or an SGML tool which could feed any of these. This step marks the text as required. Now the output file is ready for a receiving application to provide the be-

---

3 Named by Louise author Bill Zoellick, from the Paul Siebel song (sung by Gordon Lightfoot and Bonnie Raitt) that begins, "They all said Louise was not half-bad..."
behavior for those text objects, whether it's formatting a citation properly, resolving the proper page number for a cross-reference, instantiating hypertext links or executing commands to go fetch data or run an application.

Many tools can perform simple file-to-file translations or even discern paragraphs, headlines, graphics and other elementary text objects, but FastTAG does by far the best job of heavy-duty object recognition. The company also distributes a version of the Houghton-Mifflin Correctext grammar-checker which it calls Proof Positive.

---------------
The hypertext tools discussed here (with the partial exception of Folio Views) generally deal with linear documents. The other model of hypertext comes from Xerox NoteCards. That model is a set of linked nodes -- typically, cards, or little boxes, or whatever. Its most popular embodiment is in HyperCard from Apple/Claris, and, with more object-oriented programming underneath, Toolbook from Asymetrix. The other model, exemplified by Guide, is a document with links within it. In theory, you could build either in the other, or convert from one to another, but they have completely different characters.

---------------

ELECTRONIC BOOK TECHNOLOGIES' DYNATEXT (SGML compiler)

Electronic Book is the first company to implement the simple idea of a broad-based SGML hypertext compiler -- no more and no less. To use it, the builder-user must supply an SGML-compliant document including tags (typically built with an SGML editor such as SoftQuad's Author/Editor or Datalogics' WriterStation or Exoterica CheckMark, or translated from some other markup scheme). It also needs a style sheet (using an interactive graphical style editor and fill-in-the-blanks for rules and commands) to define the document's structure and desired fonts for the various text objects, button styles for the hypertext links.

Then DynaText will automatically build a hypertext document for interactive display on your "choice" of platforms, starting with Sun UNIX and Windows soon. Graphics and other non-text objects are represented as icons or displayed in separate windows, while the text remains in its own window (wrapping to fit as the window is resized). The user gets a hypertext document that he can't alter or revise, but he can fetch (through executable links) the latest data from external databases; he can also annotate the text and provide parameters to embedded commands. And of course he can browse through the document, search for words, follow links and select views.

Electronic Book offers a limited-use license for the compiler of $10,000 for 1000 units (the creation of 1000 documents, with an unlimited number of copies of each). Viewers for the resulting hypertext documents cost $500 per simultaneous viewer or less with quantity discounts.

Electronic Book was founded in July 1989 by Lou Reynolds, who learned about the importance of documentation as vp of marketing at Cadre, a leading CASE company in Providence, RI. He learned about hypertext from Andy Van Dam at Brown University, a hotbed of hypertext activity; Van Dam is now on EBT's technical advisory board. EBT's developers are also all from Brown, includ-
ing DynaText's chief architect Steven DeRose, a computational linguist and co-author of the paper cited on page 10. Reynolds financed the startup with $50,000 of his own (from Cadre stock), and has kept the company to 10 people so far. It delivered its first product early this year and made a profit in the first quarter. Reynolds wants to keep EBT as mostly a development house: The goal is to sell only to experienced customers or through resellers and consultants with SGML expertise who can help their customers prepare SGML documents. Customers at 36 sites worldwide include Westinghouse, Computer Sciences, Grumman, Boeing, Bellcore, Prime, HP, Alcatel and CERN, the Center for European Nuclear Research (the acronym is from the French).

**IBM BOOK MANAGER (Automation begins at home)**

IBM also has an industrial-strength product line in this area, BookManager, shipping since 1989. It's more flexible in what it takes than DynaText, but it goes to more work to do so; in fact, it's more or less a compiler for IBM's own "BookMaster" files, marked up in IBM's Generalized Markup Language. IBM plans to use BookManager to migrate much of its documentation to online versions (although the same revisable text files will also be used to produce printed output). "This is a strategic vehicle for IBM soft-copy manuals," says George Neu of IBM Publishing Solutions Marketing.

BookManager comprises two basic parts, a builder and a reader, and optional add-ons for tagging non-BookMaster text files and other tasks. (It's an Avalanche OEM for "TextTAGger", among other things.) From BookMaster files the Builder generates formatted text laid out for screen display at runtime according to the characteristics of the display terminal. Users can search and annotate the text and follow links, but they can't change or reformat it. The Builder operates under VM or MVS; there are four Reader options -- MVS, VM, OS/2 or DOS, which can all read the same BookManager source files.

Hundreds of customers across many industries are already using BookManager, to deliver their own documentation and procedure manuals, manage new drug applications and publish rate bases, among other things.

**TELEPRINT IDDS (Just-in-time printing)**

Teleprint is one of the oldest companies in the business, although it has changed its own business several times over the course of its eight years if existence. Caleb Avery founded the company in 1983 to offer teleprinting: "You send us your text file, and we'll lay it out and format and print it for you overnight." Drexel Burnham was the company's first customer and source of funds; it used the service to print out rapid drafts of the prospectuses of all the junk bonds and other securities it was issuing. Of course, that was just before Drexel Burnham got distracted.

Avery turned the company into a consulting firm, helping user customers handle the diversity of equipment that made it difficult to integrate and automate their inhouse publishing operations. That experience made him a big fan of SGML and standards in general. Teleprint now has 36 consultants (full- or part-time) on its staff, including SGML committee chairman Bill Davis. Much of their work is oriented to government or telephone companies; customers include Boeing, Martin Marietta and Northern Telecom.

The work with Northern Telecom, totalling $4 million over the last few years, has resulted in the Intelligent Document Delivery System, which
Teleprint markets with royalties to NT. IDDS creates and delivers page-oriented text electronically, so that only the material people actually want to see is ever printed out. It has two components: A UNIX-based Processor converts existing document text and images (formatted using PostScript or other mainstream printer languages) into Computer Graphics Metafile format, for display by the Navigator reader (for pcs, Macs or UNIX workstations). Sort of a cross between text retrieval and image systems, IDDS keeps the look and page layout of the original but also lets users search the full-text index electronically, unlike most image systems. They can also follow hypertext links and print out selections to read offline. The pages can be annotated in a separate layer, but the content and formatting are unchangeable in the read-only CGM form.

While SGML is about tagging text so it can be processed and formatted, and DynaText and BookManager format on the fly, Teleprint comes in later in the process to store and compress formatted, ready-to-print pages so they can be viewed or printed. (Typical pages average about 3K each, with about 25 percent overhead for the index.) A development system costs around $60,000, and typical orders (with workstation readers at $115 a head) run $100,000 to $200,000. There are about 5000 readers out there, at Northern Telecom, five other telephone industry customers and other sites. At NT alone, IDDS takes output from 28 different publishing and graphics applications.

"Print-on-demand is the key benefit," says Avery. "We save lots of trees." Altogether, in 1990 IDDS was used to distribute 7 million different pages electronically, saving the equivalent of 350 million printed pages. Only a small fraction of that will ever be printed out or even looked at, but before IDDS it probably would have ended up on a shelf somewhere.

**OWL’S GUIDE (Leading the way)**

OWL International was the first pc-based commercial hypertext vendor, using technology developed by Peter Brown at the University of Kent. Its product, Guide, formalized the notion of different types of links for display, as listed on page 12. OWL was established in 1985 in Seattle as an outpost of a UK development company, Office Workstations Ltd. of Edinburgh, Scotland. The goal was for the US company to publish the product and operate closer to the majority of customers, who include Boeing, Procter & Gamble and IBM. The company has sold about 20,000 copies, 4000 on Macs and the rest on pcs.

Guide, now in its third release under Windows 3.0, was like the original dBASE: a builder-user tool. The user could construct his own hypertext document, generating links manually by moving from place to place within the documents and clicking to say, in effect, link these two. The builder and user were assumed to be the same person or part of a tight group. OWL offered a runtime version, Guide Reader, only two years ago.

Now OWL is addressing a higher-end market with Guide Professional Publisher, which it will announce at the TechDoc conference in mid-August. GPP includes Avalanche’s FastTAG, GuideWriter (for converting marked-up documents), Guide (for editing, linking and customization) and GuideReader (for viewing). Builder-users can feed it ASCII, Microsoft Word, WordPerfect, DisplayWrite and other files (basically, whatever FastTAG can read) for automatic conversion into Guide documents. The package, including training, support and installation, costs $25,000.

*Release 1.0* 31 July 1991
Separately, OWL is doing some of the software work to enhance Guide for Bell Atlantic, and is also reselling the resulting product under the DocuSource name. The difference is that DocuSource is a large-scale, network-oriented preparation and delivery system, whereas GPP is a standalone system without all the bells and management/workflow-control facilities of DocuSource.

Guide is a powerful tool, but it lacks automated link-generation capabilities. The links must be built by hand, or included in the original source documents as references that can be flagged by a translator. OWL is also expanding to support multimedia, especially HyTime and the Multimedia PC specification spearheaded by Microsoft and Tandy.

"One man's link is another man's sausage."
-- Jef Raskin

SMARTEXT FROM SAMNA/LOTUS (Sculptured links)

Automated link generation is the province of SmarText ($495 for the builder; $99 for the reader) -- and of course of higher-end tools used inhouse by companies such as KnowledgeSet (see Release 1.0, 4-91). SmarText uses a number of simple algorithms to help in the generation of automatic links within a document -- and then lets the user both tune the aggressiveness of the link generation and remove extra links one by one (which is a lot easier than creating new links one by one).

Basically, it works with the source text file plus two word files -- one stopwords, and the other index terms or keywords. The index words end up in the index, and are also assumed to be assumed to be worth creating links to. The software goes through the text, looking both for single instances of the index words and other special words, and for clusters where each of those words appears with high frequency. Then the single instances of the words are linked forward to the clusters of those words, on the theory that the clusters represent explanations or deeper discussions of those words and are appropriate sections to link to. The software also looks for words that are neither so frequent as to be meaningless, nor so rare as to be irrelevant, and proposes those as other link words.

For example, in a document about Compaq, the word Compaq would appear too often to be relevant, whereas SystemPro might appear from time to time, and then very frequently in a section devoted to the SystemPro. SmarText would pick that up. "That's not very brilliant!" Juan might say to Alice. "I can understand how it works." In fact, a reviewer did say just about that. But that doesn't mean it's not worth doing automatically.

After getting the hang and feel of SmarText, a user can tune it to find more link words, or fewer, depending on his preference. You can also manually add words to the stop word or keyword list. And of course you can manually delete extraneous links, add new ones, and you could link a phrase such as "high-end systems" to the SystemPro section as well. In a future release SmarText will handle synonyms automatically, but it doesn't yet. (It will also support Microsoft's Object Linking and Embedding soon.)
SmarText currently works with a variety of standard word-processor files such as Ami Pro (naturally), Microsoft Word, ASCII and Microsoft's Rich Text Format (soon), and keeps separate graphics files for images. It doesn't mark them up directly, but works by adding a sort of shadow layer of links, views and other features. The advantage is that the source text is kept unchanged, and can be edited in its original format; the disadvantage is that although altered texts can be reprocessed automatically (with the revised keyword and stopword lists), some manual work performed removing extraneous links is lost when the revised file is reprocessed. However, the developers went to considerable lengths to be able to save manually created links by saving the surrounding text and reinstating the link. (Of course, if the new edit deletes a section containing a link, the link is lost, but otherwise the process works reasonably well.)

From our perspective, SmarText could be enriched considerably with the addition of SGML and objects. The keywords, at a minimum, defined as objects, could be formatted differently. (The application does this, but not as simply or cleverly as if they were defined as objects.) More interestingly, it would be easier to derive the locations of clusters, which are now determined by paragraphs; defined sections would provide greater accuracy in finding clusters, and more intelligent linking. In addition, defined headlines for those paragraphs would provide extra clues for the linking: It's a pretty safe bet that the sections following a headline reading "word word word SystemPro word word" [where "word" is unmarked text] has something to say about the SystemPro. You could give words in a headline extra weight, for example, and link to the beginning of the right section.

Another obvious enhancement would be to link SmarText to Notes, where it could use Notes structures to do much of what we just described -- unfortunately in a proprietary way. Some day, we hope, Notes will have facilities for conversion into and out of SGML (to say nothing of full SGML support). Perhaps that's a promising opportunity for a third party.... Says Mohammadioun, "If the world standardized on anything it would be a benefit to us."

SammarText

SmarText was developed from a semi-independent company, Big Science, funded and half-owned by Samna, much the same way Notes was developed by Iris Associates with sponsorship from Lotus. Big Science was founded in 1988 by three engineers from the Lockheed Pilots Associate project who approached Samma founder Said Mohammadioun for funding. They agreed that the group would develop the product, and Samna would market it.

As it happened, the product was ready for launch and shipped last October, just about the time that Lotus acquired Samna, leaving Mohammadioun with more time to get personally involved. Over the past six months, instead of promoting the product vigorously with a confused message, he has instead shown it directly to a dozen or so sizable companies, including Fidelity Management and some pharmaceutical companies wrestling with how to handle their FDA filings and doctor's reference materials, soliciting both trial purchases and feedback. (The Windows User Group also uses it for its newsletter, a nice reference account.) The feedback has helped Samna refine the marketing strategy. Accounts interested in multimedia and presentations tend to use Guide, and probably should. On the other hand, SmarText is a lot closer to Folio Views in its concentration on content and text over form.

Release 1.0
31 July 1991
FOLIO VIEWS (First distribute the razor blades...)

Folio Views (see Release 1.0, 3-90) is an ideal tool for document assembly from a bunch of disorganized chunks, whereas SmarText is best for taking a linear document and finding the connections between the parts. The assumption is that with Folio an author is assembling the parts into an "infobase," whereas with SmarText a second person may be automatically structuring a document submitted by another author. Of course, both can do either task, and Views is also good for assembling separate documents submitted by multiple authors. (But to classify them and perhaps detect redundancies, or inconsistencies, IZE, below, is the most powerful automated tool.) The Views professional publisher tool costs $695, and a personal tool, for annotation and organization but not creation of infobases, costs $295.

Views also has a small, simple, inexpensive reader version, generally OEMed and bundled with other products, which makes it ideal for delivering documentation with packaged pc software. In fact, it has by far the largest installed base of copies -- about 10 million of them bundled with Novell NetWare. Obviously, not all of them are actively used, but Folio says many of its largest customers (including a Hartford insurance company 3300 with 3300 copies of the tool) first tried out Views as part of NetWare. Folio Views is also the delivery vehicle for Ziff-Davis's new Magazine Rack CD-ROM product.

Folio Views has integrated full-text search, which Guide still lacks. Although it doesn't build the links automatically like SmarText, it can easily find clusters and make it easy for a user to build his own links. In the end, we believe Folio is better for online use of chunked data, whereas SmarText is better for documents that may be printed out as much as they are used online. (The links SmarText creates can be printed as cross-references, as in "See page 4.") Folio has more of the cards feel, while a SmarText document is linear; there's one basic form from which SmarText views are derived. (Its automatic generation of views involves assembling the clusters about a particular word or several words into a focused subset. Of course, a user can build views manually by selecting and ordering the sections he wants.)

IZE FROM RETRIEVAL DYNAMICS INC. (Big trees from a little algorithm)

We fell in love with IZE years ago, even before it was acquired by Persoft in 1987 (see Release 1.0, 5-87). The product finds word clusters akin to those SmarText looks for, but it doesn't take no for an answer; every text item is classified by some word until there are no twice-used words left. IZE uses a simple, now patented algorithm: "Find the most common word in the text (except for stop words), and divide the text into two buckets of paragraphs or whatever chunks you're using, one with the word and one without. Do the same again and again." You end up with a tree structure of text items that usually has a surprising degree of relevance. You can tweak it by adding or removing words form the stop list. This simple algorithm allows you to generate a powerful hierarchy classifying your texts, so that you have a map or tree instead of just some links. In short, it generates a structure for the text, rather than a set of links.

While other text classifiers can tell you how relevant two documents are to each other, and other tools can link related sections, they can't easily handle the relationship of more than two (which requires a representational of multi-dimensional space). But IZE can classify a whole set of texts in a...

Release 1.0 31 July 1991
two-dimensional tree. Since IZE is the most automatic of the pc-based text- 
classification tools, it's best suited for automatic maintenance of text on a 
server, while the others are better suited for development by one person at a 
time, with distribution over a network. (Verity's Topic, by contrast, lets 
users create a tree manually to model a particular content domain and then 
classify text automatically according to the model; see Release 1.0, 3-90.)

Persoft founder Ed Harris has now spun off himself and the IZE division into 
a new company called Retrieval Dynamics Inc., which is still sharing the Per- 
soft offices until it gets outside funding. Harris is working with developer 
Paul Kleinberger on a network version that would manage access, as opposed to 
the current version, which simply lets users share files on a server.

ACTIVE DOCUMENTS TECHNOLOGY FROM INTERLEAF (Power of objects)

Interleaf's Active Documents technology, an extension to its Interleaf 
document-processing system, is the exemplar of the power of combining text 
and true objects: a fully object-oriented text-creation and management sys- 
tem, fully described in Release 1.0, 3-90. Because it is written in LISP 
(the base product is in C) and is fully object-oriented, the technology is 
completely extensible and can be made to interact with any other system using 
interprocess communications. Its text objects have their own behavior, and 
can also interact with each other rather than be controlled by, say, a for- 
mattting program. That is, the text objects in most SGML systems rely on ex- 
ternal applications for their power (unless you have embedded executable 
code), but the Interleaf objects contain their own behavior, bound dynamical- 
ly at runtime within the Interleaf environment.

There is nothing in particular that you can do only with Active Documents, 
but it can be extremely awkward and clumsy to write the software to do so. 
The Interleaf system makes it (relatively) easy both to modify behavior by 
changing parameters through dialogue boxes, and, for professional developers, 
to create new objects by modifying existing ones with new behavior.

Unfortunately, however, this power isn't easily transferable to other envi- 
rionments. You can translate Interleaf documents into and out of SGML, yes, 
but it's a little like translating a database file and the data structure 
catalogues into wp format; you lose all the power when you do so. The only 
way to get the power back is to put the data back into the engine. The docu- 
ments aren't self-running object-oriented systems, but are dependent on the 
presence of the Interleaf class library and operating environment. The real 
benefit is the built-in functionality, not the ability to call outside apps.

NEW TECHNOLOGY FROM PAGES (We want one!)

The Pages product line is one of the most impressive tools we've ever seen... 
Of course, the bad news is that it won't be available until next year (and 
this section is purposely a little vague; sorry!). It's also the application 
that shows why you'd buy a NeXT machine. Watching it shuffle text objects 
around 16 pages automatically as fast as a spreadsheet can recalc is like 
watching three days of frustrating work happen in seconds. (We know!) Like 
Interleaf's Active Documents, the Pages line is fully object-oriented, writ- 
ten in Objective-C (less of a standard but more truly object-oriented, with 
dynamic binding, than C++) and NeXT's Interface Builder, along with Pages' 
own proprietary document-oriented rule/constraint language.

Release 1.0 31 July 1991
The product line comprises both builder and user modules. The user module helps end-users enter text and lay out pages while it automatically enforces design constraints about fonts, styles and placement of objects, including complex rules about relationships between text and objects, page balance and the like. The design constraints come either pre-packaged for certain looks, or can be modified through the builder modules by experienced designers who can share their taste and wisdom through this medium. "Make it look like Upside Magazine, except..." is the kind of request Pages can handle.

A user invokes a design model which incorporates the expertise of a professional designer, yet allows the user to provide parameters and select (within constraints) such attributes as fonts and other formatting for the text objects and their locations on a page. Do you want two columns or three? Where do you want the page numbers (odd or even)? And so forth.

The foundation of the Pages line is a full-fledged object-oriented development environment -- a class library of text objects which can be subclassed or extended. Pages is using a prototype of it to develop the components of the product line. The goal is to build a user rule-tool interface that is easier to use than Objective-C or the Pages rule language, so that graphic experts, rather than Objective-C experts, can add classes and behavior rules. Builder-users would be able not just to modify but to create design models. This is where the real power of Pages' tools lies. A power user can define and create new kinds of text objects, and construct rules and constraints regarding their use and appearance.

Because the text objects are active objects with behaviors and inheritance, it is relatively easy to define new ones. For example, defining a third-level headline is easy; defining a specific type of list item -- for example, with the first phrase highlighted -- is a little harder. (See page 10.)

In fact, that first phrase of each list item could itself be a text object; a formatting rule might dictate that it should be followed by a comma -- or perhaps a dash, or whatever a user specifies. A constraint could be that the dash should be followed by a full sentence, whereas the comma takes a modifying clause. (How do you tell whether it's a modifying clause or a full sentence? That's an interesting question, which would require a smart user or the kind of parsing a grammar-checker does. Any volunteers?)

A less challenging example is layout rules, ranging from no-more-than-one-picture-to-a-page to how diagrams should be adjusted to remain as close as possible to references to them in the text. For example, as every publisher knows, there are times when you refer to a page-24 illustration on page 23, a page-turn early. But if you put the illustration on page 23, the text reference would move forward to page 24. How do you want to handle that?

Watching the Pages prototype go through its paces is tremendously exciting. You could tell it which way to resolve the page 23-24 question, or you could tell it to flash an error message -- a friendly one, of course, suggesting that you rearrange the text. The user could decide to move some sections around, and watch the whole laid-out document -- up to 16 miniaturized but clear pages on a NeXT display -- rearrange itself, observing all constraints. Then the reference could appear on page 22, facing the picture on page 23.

The challenge for Pages is to build a truly friendly rule specification interface and a powerful but easy-to-use object editor. These will allow a
graphic designer to express a huge amount of expertise and taste and ultimately creativity for reuse by less skilled people.\(^4\) We're looking forward to it -- and you should too, as a reader of Release 1.0!

Pages has its own rule language and uses Objective-C to define the behavior of its objects. For the moment, it relies on users to specify the text objects, but there's no reason (with a little bit of work) that it couldn't accept SGML files and DTDs to specify the objects in a file. Then, except for tweaking, it could lay out a document automatically. SGML specifies the objects; Pages describes and automates their behavior and the application of formatting and layout rules. Pages was founded in 1990 by Mike Parker, who also founded font firm Bitstream and The Company (acquired by LaserMaster), a developer of intelligent rasterizer tools.

ACCURATE INFORMATION & ONTOS (Text objects any way you want them)

Accurate Information Systems, Inc., is a 100-person company with offices in New Jersey and the Washington, DC, area, mostly near military locations. The company made $18 million in revenues last year, much of it from designing and testing CALS-oriented systems and standards for managing documentation. It provides support for the Army's CALS Test Bed in Fort Monmouth, NJ, developing and debugging demonstration projects using CALS tools such as those described in this newsletter, interoperating across a variety of hardware and software environments. (It also does plain old office automation and is soon to open a Novell Authorized Training Center, among other things.)

Its most interesting project (from our perspective) involves storing documentation components in an object-oriented database, Ontos from Ontos Corp. (formerly Ontologic) for re-use within a variety of applications including documentation systems. The same components can then be assembled, using DTDs and various selection criteria, into a variety of different documents with subsets of the information. Specifically, the demonstration project showed the generation of a maintenance information module of an Army technical manual based on information from a maintenance allocation chart (MAC), a sort of spreadsheet of components and related repair functions and equipment. The MAC (in this case, referring to an M1A1 tank) can now be updated through the object-oriented database, and vice versa. (See next page.)

The underlying information is the same, but the presentation is radically different -- determined both by a DTD and a POSI (for formatting). The ultimate goal, of course, is that the same information for the same equipment can be reused across service boundaries, each of which has its own documentation formats but could use the same SGML source files with its own DTDs and formatting instructions. This is the basic goal of the JUSTIS project (page 3).

\(^4\) We can imagine such a tool with some rule-by-example capabilities -- beyond what Pages is now promising -- so that a builder-user could simply give the system some examples and let it derive the rules. This would use the same kind of pattern-recognition capabilities as Apple's Eager system, described in our 5-91 issue, or perhaps an extension of Avalanche's object-recognition techniques.
As shown in the flowchart across, the maintenance allocation chart (A) feeds the OODB (modeled in B) in step 1. In step 2, a subset of the data is assembled according to a DTD. Then it is edited by a writer (3) and formatted (4) into an intelligible page for a manual (C).

### Table: Maintenance Allocation Chart

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Number</td>
<td>Component/Assembly</td>
<td>Maintenance Function</td>
<td>C</td>
<td>O</td>
<td>F</td>
<td>H</td>
<td>D</td>
<td>Tools and Equipment</td>
<td>Remarks</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>Tank</td>
<td>INSPECT</td>
<td>0.10</td>
<td>0.50</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>ENGINE</td>
<td>SERVICE</td>
<td>0.40</td>
<td>1.24</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>010</td>
<td>ENGINE</td>
<td>TEST</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>010</td>
<td>ENGINE</td>
<td>REPLACE</td>
<td>0.30</td>
<td>1.24</td>
<td>6,9,10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>010</td>
<td>ENGINE</td>
<td>OVERHAUL</td>
<td>21.40</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>010</td>
<td>ENGINE MOUNTS</td>
<td>INSPECT</td>
<td>0.30</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>010</td>
<td>ENGINE MOUNTS</td>
<td>REPLACE</td>
<td>0.80</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>010</td>
<td>CYLINDER HEAD</td>
<td>INSPECT</td>
<td>0.30</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>010</td>
<td>CYLINDER HEAD</td>
<td>REPLACE</td>
<td>3.20</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>010</td>
<td>CYLINDER HEAD</td>
<td>REPAIR</td>
<td>2.70</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>010</td>
<td>PERISCOPE</td>
<td>INSPECT</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>010</td>
<td>PERISCOPE</td>
<td>REPLACE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>010</td>
<td>PERISCOPE</td>
<td>REPAIR</td>
<td>1.90</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>010</td>
<td>PERISCOPE</td>
<td>SERVICE</td>
<td>0.10</td>
<td>1.24</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Diagram: Periscope Washer Fluid Reservoir

The diagram illustrates the Periscope Washer Fluid Reservoir. The notes indicate the various components and their connections. The numbers correspond to the steps in the maintenance process.

**Release 1.0**

31 July 1991
RESOURCES & PHONE NUMBERS

Rita Knox, Accurate Information, (908) 389-5550; fax, (908) 389-5556
Haviland Wright, Avalanche Development, (303) 449-5032; fax, (303) 449-3246
Marc Stigler, Chris Hibbert, Autodesk (Xanadu), (415) 332-2344
Jeff Beegle, Bell Atlantic (Docusource), (201) 649-6088; fax, (201) 649-4513
Lou Reynolds, Electronic Book (DynaText), (401) 421-9550; fax, (401) 421-9556
John McFadden, Cindy Sprang, Exoterica, (613) 722-1700; fax, (613) 722-5706
Steve Brown, Datalogics, (312) 266-4400; fax, (312) 266-4473
Pat McGloon, DEC, (603) 884-3982; fax, (603) 884-5284
Curt Allen, Folio Corporation, (801) 375-3700; fax, (801) 374-5753
George Neu, IBM (Book Manager), (714) 241-3228
Chuck Cooper, Frosty Gordon, IBM, (303) 924-7377; fax, (303) 924-9153
Steve Pelletier, Dave Weinberger, Interleaf, (617) 290-0710; fax, (617) 290-4943
Tom Rolander, KnowledgeSet, (408) 649-4193
Said Mohammadioun, Lotus (Samma), (404) 851-0007 x 200; fax, (404) 256-4104
Frank Ingari, Ontos, (617) 272-7110
Bill Nisen, Alister Gibson, OWL International, (206) 747-3203; fax (206) 641-9367
Phil Cook, OWL International, 44 (31) 557-5720
Bruce Webster, Pages, (619) 492-9050; fax, (619) 492-9124
Ed Harris, Retrieval Dynamics Inc. (IZE), (608) 273-6000; fax, (608) 273-8227
Yuri Rubinsky, SoftQuad, (416) 239-4801; fax, (416) 239-7105
Caleb Avery, Teleprint, (303) 792-3100 or (800) 543-6899; fax, (303) 792-3757
Bill Davis, Teleprint Technical Services, (703) 370-5550; fax, (703) 370-5551

For further reading:

"Standards and the electronic publishing industry," speech by Teleprint's Bill Davis to the Xplor conference, November 1990. Not just another panegyric, but a useful history and rationale for publishing standards by a user (at the IRS) who helped create them.

Hypertext and hypermedia handbook, edited by Emily Berk and Joe Devlin, McGraw-Hill/Armadillo Associates, 1991. Especially a section by Thomas C. Rearick on "automating the conversion of text into hypertext." The section is a bit of a plug for SmarText, but it deserves it. Overall, there's lots of good material in here.


TAG>, the bi-monthly SGML industry newsletter, edited by Dale Waldt and produced by Graphic Communications Association. A handy guide to politics and progress. (716) 671-7780, x 245.

"Digital Technical Journal," Winter 1990. This issue focuses on DEC's Compound Document Architecture, which we didn't have space or time to cover in this issue.

Release 1.0 31 July 1991
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 11-13</td>
<td>*GeoCon/91 - Cambridge, MA. Sponsored by Softletter. An international product showcase for European, Canadian, Asian and Latin American developers who seek U.S. publishing or partnership contacts. Call Jeff Tarter, (617) 924-3944.</td>
<td></td>
</tr>
<tr>
<td>August 19-23</td>
<td>TechDoc '91 - Seattle. Sponsored by Graphic Communications Association. See all the SGML vendors and others described in this issue. Call Joy Blake, (703) 519-8177.</td>
<td></td>
</tr>
<tr>
<td>August 19-23</td>
<td>SCO Forum91 - Santa Cruz, CA. Sponsored by The Santa Cruz Operation. Call Zee Zaballos, (408) 425-7222.</td>
<td></td>
</tr>
<tr>
<td>September 4-6</td>
<td>UNIX Open Solutions - San Jose. Sponsor: Interface Group. Keynotes by Scott McNealy, Sun; Doug Michels, SCO. Call Elizabeth Meagher, (617) 449-6600 or (800) 325-8850.</td>
<td></td>
</tr>
<tr>
<td>September 5-6</td>
<td>Integrating image and information processing - Washington, DC. Sponsor: DCI. Call Karyn Green, (508) 470-3880.</td>
<td></td>
</tr>
<tr>
<td>September 10-12</td>
<td>DataStorage91 - San Jose. Sponsored by Freeman Associates and Disk/Trend. Call Darlene Plamondon, (408) 554-6644.</td>
<td></td>
</tr>
<tr>
<td>September 11-14</td>
<td>Software Publishers Association annual conference - Orlando. Sponsored by SPA. Call Ken Wasch, (202) 452-1600.</td>
<td></td>
</tr>
<tr>
<td>September 15-19</td>
<td>*EastEurOOPe '91 - Bratislava, Czechoslovakia. Sponsored by JOOP, ParcPlace, Xerox, Digitalk, Software Slusovice, Kancelarske Stroje, others. With Adele Goldberg, Krysten Nygaard. Contact: Augustin Mrazik or Peter Mikulecky, 42 (7) 724-826; fax, 42 (7) 725-882; e-mail: <a href="mailto:eeoop91@mff.uniba.cs">eeoop91@mff.uniba.cs</a>.</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Event Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>September 25-27</td>
<td>*Second European conference on computer-supported cooperative work - Amsterdam. Organized by the University of Amsterdam. Call Mike Robinson or Liam Bannon, 31 (20) 525 1250/1225; fax, 31 (20) 5251211; e-mail, <a href="mailto:Bannon@learn.ucd.ie">Bannon@learn.ucd.ie</a>; or Charlie Grantham, 1 (415) 370-1744; <a href="mailto:cegrant@well.sf.ca.us">cegrant@well.sf.ca.us</a>.</td>
<td></td>
</tr>
<tr>
<td>Sept 30-Oct 1</td>
<td>Virtual Reality conference - San Francisco. Sponsor: Meckler Corp. Call Marilyn Reed, (203) 226-6967 or (800) 635-5537.</td>
<td></td>
</tr>
<tr>
<td>Sept 30-Oct 4</td>
<td>*Seybold Conference - San Jose. The leading event in the computer publishing community. Sponsored by Seybold Seminars/Ziff. Call Kevin Howard or Beth Sadler, (213) 457-5850.</td>
<td></td>
</tr>
<tr>
<td>October 6-11</td>
<td>*OOPSLA '91 - Phoenix. Sponsored by ACM. Call John Richards, (914) 784-7731.</td>
<td></td>
</tr>
<tr>
<td>October 16-18</td>
<td>EDUCOM '91 - San Diego. Sponsored by University of California at San Diego. Speakers include Sheryl Handler, Bill Joy. Call Diane Balestri, (202) 872-4200.</td>
<td></td>
</tr>
<tr>
<td>October 21-25</td>
<td>*Comdex - Las Vegas. So wonderful they couldn't wait until November? Whatever the reason.... Sponsored by Interface Group. Call Elizabeth Moody or Dick Blouin, (617) 449-6600.</td>
<td></td>
</tr>
<tr>
<td>November 10-13</td>
<td>**Second East-West High-Tech Forum - Warsaw (Prague in 1992). Sponsored by EDventure Holdings. With a roster of serious-minded entrepreneurs and vendors from East and West. Don't just come to listen to advice; come to mingle with the people making it happen. Call Daphne Kis, 1 (212) 758-3434 or fax (212) 832-1720; MCI Mail: EDventure, 443-1400.</td>
<td></td>
</tr>
<tr>
<td>February 23-26</td>
<td>**EDventure Holdings PC (Platforms for Computing) Forum - Tucson, AZ. You read the newsletter; come meet the community and try its tools. Call Daphne Kis, (212) 758-3434.</td>
<td></td>
</tr>
</tbody>
</table>

Please let us know about any other events we should include. -- Denise DuBois

Release 1.0 is published 12 times a year by EDventure Holdings, 375 Park Ave., New York, NY 10152; (212) 758-3434. It covers pcs, software, CASE, groupware, text management, connectivity, artificial intelligence, intellectual property law. A companion publication, Rel-EAST, covers emerging technology markets in Central Europe and the Soviet Union. Editor & publisher: Esther Dyson; associate publisher: Daphne Kis; circulation & fulfillment manager: Robyn Sturm; executive secretary: Denise DuBois; editorial & marketing communications consultant: William M. Kutik. Copyright 1991, EDventure Holdings Inc. All rights reserved. No material in this publication may be reproduced without written permission; however, we gladly arrange for reprints or bulk purchases. Subscriptions cost $495 per year, $575 overseas.

Release 1.0            31 July 1991
Please enter my subscription to Release 1.0 at the rate of $495 per year in the U.S. and Canada. Overseas subscriptions are $575, airmail postage included. Payment must be enclosed. Multiple-copy rate on request. Satisfaction guaranteed or your money back.

Name
Title
Company
Address
City State Zip
Telephone

How did you hear about Release 1.0?

Payment enclosed.

American Express # Expires

Please send me additional information on your multiple-copy rate.

Please fill in the information above and send to:

EDVENTURE HOLDINGS INC.
375 PARK AVENUE, SUITE 2503
NEW YORK, NY 10152

If you have any questions, please call us at (212) 758-3434.

Daphne Kis 
Associate Publisher