Journal of Computing in Higher Education
Winter 1990, Vol. I (2), 3-26.

What is Text, Really?

Steven J. DeRose
praXis

David G. Durand
Brandeis University

Elli Mylonas
Harvard University

Allen H. Renear
Brown University

ABSTRACT

THE WAY IN WHICH TEXT IS represented on a computer affects the kinds of uses to which it can be put by its creator and by subsequent users. The electronic document model currently in use is impoverished and restrictive. The authors argue that text is best represented as an ordered hierarchy of content object (OHCO), because that is what text really is. This model conforms with emerging standards such as SGML and contains within it advantages for the writer, publisher, and researcher. The authors then describe how the hierarchical model can allow future use and reuse of the document as a database, hypertext, or network.

ALTHOUGH COMPUTING VISIONARIES have spoken of an almost magical future world of accessible information and communication, years of progress have still failed to realize Alan Kay's image of "piloting a jet plane through information space." The authors emphatically believe that the computer has immense potential to assist people in general, and scholars in particular, in their intellectual work. This assistance will extend across such activities as researching, thinking, and writing, as well as recording and organizing information. However, current software technology has not yet lived up to this potential. In this article we examine how the *model of text* embodied in a computer system can limit a scholar's ability to accomplish desired ends with text, and we argue that by adopting a more sophisticated model (which we believe better reflects the realities of text and documents), scholars can begin to gain some of these long-sought but seldom found benefits.[1]

Too often, when seated before a computer or terminal, the dominant feeling is just that of faster and easier typing; most of the conceptual and even motor operations are closely modelled after those of carbon ribbon days past. This alone shows that the sophistication of the computer is not fully utilized; but worse than the lack of qualitative improvements over the past is the fact that some old advantages have actually been lost. Consider these disillusioning problems that have never been found in the world of pens and ink:

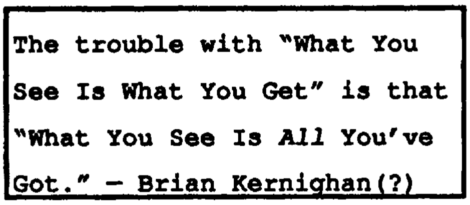
- we can't share documents (or "files") with colleagues because they are using different word processors;
- publishers can't typeset directly from our disks, but must re-key everything, introducing new errors;
- the printer across campus produces different results (or often no results at all), as compared to the one down the hall;
- the huge amount of text (papers, research notes, book lists, etc.) accumulated over years of working on-line, still cannot be navigated effectively: We may know that a quote is *somewhere* on our disk, but still be unable to find it.

The authors contend that both the pedestrian nature of most text processing, and these problems with printing, portability, and information retrieval, have a common source. It is not a lack of computing power, memory, or pixels; nor is it that software applications are not ingenious or "friendly" enough. The problem is much more fundamental, and has to do with the *organization* of most computer stored text. No hardware improvements or programming ingenuity can completely overcome a flawed representation.

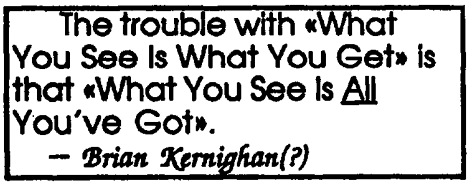
Because text has the longest history of explicit organization among forms of communication, and is a key concern of many scholars, we have chosen it as our focus here. However, we believe that these principles apply, *mutatis mutandis*, to other media both ancient and modern.

OHCO: WHAT TEXT REALLY IS

THE COMPUTER REPRESENTATION of a document should reflect what really *is*. Thus, the first question is "What is a document?" One way to approach this question is to consider the question of *essentials*: What is it which, if changed, makes a document essentially different, and what is it which can change, yet a document remains "the same?"



```
The trouble with "What You
See Is What You Get" is that
"What You See Is All You've
Got." - Brian Kernighan(?)
```



The trouble with «What You See Is What You Get» is that «What You See Is All You've Got».
— *Brian Kernighan(?)*

In a reasonable sense, the two document fragments shown above are "the same." For typical purposes, the meaningful units, from words on up, are the same. If these change, however, the document becomes different in a deeper way than the superficial adjustments of typography we have illustrated. The distinction is largely one of *form* versus *content.*

The essential parts of any document form what we call "content objects," and are of many types, such as paragraphs, quotations, emphatic phrases, and attributions. Each type of content object usually has its own appearance when a document is printed or displayed, but that appearance is superficial and transient rather than essential — it is the content elements themselves, along with their content, which form the essence of a document. When mnemonic names for these objects are specified, a document is said to include "descriptive markup."[2]

Most content objects are contained in larger content objects, such as subsections, sections, and chapters. In the above example, the paragraph contains two quotations, the second of which contains an emphatic element. Generally, smaller content objects do not cross the boundaries of larger ones; thus a paragraph will not begin in one chapter and end in the next. For this reason, the structure of a document is a hierarchical one, like a tree or taxonomy.

Smaller content objects that occur within a larger one, such as the sections within a chapter, or the paragraphs, block quotes, and other objects within a section, occur in a certain order. This ordering is essential information, and must be part of any model of text structure.[3]

Combining these essential elements, we can describe a text as an "ordered hierarchy of content objects," or "OHCO." This is the basic model of text for which we will argue in the remainder of this article. After showing why less sophisticated models are inadequate (despite having been used as the basis for many computer tools), we will describe a number of concrete advantages which authors and particularly scholars can derive from the incorporation of such a model into software systems. The use of hierarchically structured representations of the content objects of a document is the most powerful form of descriptive markup.

There is evidence for the reality of this model. Briefly put,

- this model reflects linguistic discourse structure; texts are linguistic objects, so this is an appropriate scheme;
- many published texts already express some of this structure implicitly — for example, via tables of contents;
- manuals of style describe rules in terms of content objects — for example, a journal may specify how "block quotations" are to be formatted, showing that such units are meaningful ones to authors and editors;
- one can readily elicit many of the elements of concern in the OHCO model merely by pointing to parts of a book and asking someone what their names are;
- as Figure 1 suggests, it is natural to use this model in helping children understand and create documents.

A final point regarding the effectiveness of this model is that it survives not only changes of layout, printing technology, and so on, but

This example uses a slightly simplified version of the SGML standard for document preparation to show how one might enter a short letter and what it might look like printed out on paper.

```
<friendly_letter>
<greeting>Hi Mom</greeting>
<body>
   <pp>I've been having a good time here at camp, but the
food is awful. Do you think you could send me something
nourishing like chocolate chip cookies? Mr. Fenster fell into
the lake last week. Boy, did he look funny!</pp>
   <pp>By the way, I've spent a little more money than I
thought I would at the camp store buying this neat new
computer. Do you think you could send me some more? Thanks a
lot.</pp>
</body>
<signature> Scooter</signature>
</friendly_letter>
```

What the letter looks like

June 3, 1987

Hi Mom!

I've been having a good time here at camp, but the food is awful. Do you think you could send me something nourishing like chocolate chip cookies? Mr. Fenster fell into the lake last week. Boy, did he look funny!

By the way, I've spent a little more money than I thought I would at the camp store buying this neat new computer. Do you think you could send me some more? Thanks a lot.

Sincerely,

Scooter

Figure 1. A Friendly Letter

even translation. Although all of a document's words and most of its syntactic structure may change when it is translated into another language, the OHCO structure of the document is likely to remain intact. Re-casting the previous example, the structure remains:

> La problema de "Lo que vea,
> tenerá" es que "Lo que vea,
> es *todo* que tenerá." — Brian
> Kernighan(?)

ALTERNATIVE MODELS

DESPITE THE MANY ADVANTAGES OF THE OHCO MODEL, computer software for managing text has most commonly been built using less sophisticated models. This section will consider several alternative models: text as bitmap, text as a stream of characters, text as formatting instructions, text as page layout, and text as a stream of content objects.

TEXT AS BITMAP

SUPPOSE ONE SCANS A DOCUMENT using an image scanner; this produces a graphic image of the page. While a human reader can recognize letters by examining a picture drawn on the screen, just as with handwritten letters, there is no unambiguous computer-usable indication in the image file about which characters are represented on the page. Because of this, users of the computerized text cannot reliąbly search for words within the text, modify it, or re-format it.

A bitmap is ideal for *some* computational purposes, such as storing and displaying facsimiles of manuscripts for use in paleography and related fields. No one, however, would try to do word processing with a bitmap representation. Even archival systems which store facsimiles of paper documents must associate some text or descriptive matter with each page image, and it is this which is used for indexing, searching, and so on.

TEXT AS A STREAM OF CHARACTERS

THIS APPROACH is commonly used for mailing files through computer networks; the only encoding of structure is via insertion of blank spaces and carriage returns.[4] Since characters are explicitly encoded, the grossest shortcomings of the bitmap model are overcome. The inclusion of at least the rudimentary markup provided by blank spaces and punctuation allows a few units such as words and sentences to be located fairly reliably. Nevertheless, the essential content and structure of a document is much more than its characters, and most of this remains unavailable for processing; therefore all the problems of the last model apply to this one as well. It is virtually impossible to do anything but search for and change words. For instance, only if the structure of the document is indicated can one ask the computer for such things as a list of all block quotations attributed to Northrop Frye in a particular section of an on-line document.

TEXT AS CHARACTERS AND FORMATTING INSTRUCTIONS

FOR THE JOB OF PRODUCING FORMATTED PAPER OUTPUT, the typical word-processor file is a considerable improvement over the bitmap and the stream of characters models. Such a file contains a series of characters and spaces, plus occasional instructions for typographic processing. As with the previous model, each letter of content is explicitly represented. An editing program can locate words, delete characters, perform global changes, and support the ordinary activities of word processing; additional programs like automatic spelling verifiers, thesaurus programs, and indexing programs can also be used.

However, consider locating the next poetry quotation or the next equation. These are natural operations, on objects that are of interest to authors; yet they are not possible, because there is no explicit indication of what part of the text is poetry, or what part is an equation. Formatting commands may give some clues, but they are unreliable for several reasons:

- a (possibly long) sequence of formatting commands must be remembered or reconstructed to pick out just one kind of appearance;

- in many programs, a given appearance may be obtained in many different ways, failing to provide any unique identifying features for that appearance;
- the writer may not have formatted all content objects of a given type in quite the same way, and so some will be missed by a search;
- the writer may have formatted more than one type of object the same way, making them indistinguishable by appearance.

Compensating for problems like these is the sort of irrelevant computation the computer should free us from. The computer should similarly free a publisher's compositor who wishes to systematically alter formatting for classes of content objects, but it cannot because the necessary descriptive information is *not there*. The information which *is* present — formatting information — is inessential: it has only to do with a particular design style, a particular text processing program, and a particular output device.

TEXT AS PAGE LAYOUT

THIS WAY OF REPRESENTING A DOCUMENT, perhaps best exemplified by PostScript^{TM}, is hierarchical, and is widely used because of its similarity to familiar paper methods.[5] A book can be divided into pages; a page into the header, the main text area (perhaps with several columns, embedded pictures, etc.), an optional footnote area, and a footer. However, even this model fails to provide the kind of text handling needed by authors and scholars. How can one find equations, poetry quotations, lines of verse, and the like? The problems are exactly the same sort of those encountered with formatting instructions. The page layout model is useful for describing the position of text on the page, but it cannot readily be converted for other uses. As we shall see, the OHCO model can be easily mapped into these other models.

TEXT AS A STREAM (NOT HIERARCHY) OF CONTENT OBJECTS

THIS MODEL DIFFERS from the OHCO model only in that the content objects, indicated by descriptive markup, occur in sequence rather than as a hierarchy. Any individual piece of text can only be part of one object, so higher level content objects such as chapters and sections cannot be represented directly. This model is now being adopted by many word processors. For example, Microsoft Word^{TM} for the Macintosh^{TM} permits defining specific types of content objects as "styles." Unfortunately these styles cannot apply to anything other than a paragraph: smaller or larger objects with hierarchical relationships cannot be defined (such as the many sub-objects of bibliography entries, and higher-level units such as chapters). The MS-DOS^{TM} version supports one level of "character styles" as well, but this is an inadequate and *ad hoc* improvement.

Hierarchical markup is inevitable at some level, because containment naturally describes the structure of most texts. Therefore, a stream model must treat certain structures, such as bibliographies, footnotes, and outline views, unnaturally. Non-hierarchical markup is obfuscatory for certain structures. As a simple example, there can be no "lists," only list items; and no "chapters," only chapter headings. A display or formatting program cannot unambiguously manipulate the larger units, because they are left implicit. For example, even though "list-header" tags happen to be present, it is ambiguous whether the following sequence of objects is two lists, one after the other, or one list, with an embedded list as its third item:

```
<listheader>Advantages to be sought
<listitem>Perspicuity
<listitem>Portability
<listheader>Other advantages
<listitem>Support for multiple authors
<listitem>Ease of re-using data
```

Largely because of these ambiguities, non-hierarchical markup systems tend to be *uneconomical.* Special "embedded list item," "blockquote embedded paragraph," "continued paragraph," and similar tags proliferate, because formatting and other operations on the text cannot refer to structural context. In a hierarchical system, the number of distinct tags can be vastly reduced without loss of function. Systems which do not support hierarchical structures are painfully limited in capabilities and interface design; the larger the project, the more troublesome such limits become.

SGML: A STANDARD FOR CREATING OHCOS

SGML IS THE ISO "Standard Generalized Markup Language." It defines a powerful language for describing and documenting hierarchically structured documents of arbitrary complexity with simple character stream files.[6] It *does not* specify a particular set of content object types or "tags," but rather provides a way for declaring which tags are to be used, along with their permissible relationships. For documents of fixed form such as dictionaries and reference works this can be a great help in establishing a consistent structure. Even in the case of more loosely structured material such as literary texts, the existence of a precise description of the document structure can be of use in analysis. For these texts SGML may be more an aid to the scholar than the author.

SGML defines a document in terms of its OHCO structure: it does not directly specify how to format or process a document, but describes a hierarchical document structure with mnemonic names for the content objects of the data. Thus, it does not prejudice whether a document is to be treated as a database, a word-processing file, or something completely different. It is important to note, however, that this independence does not prevent an SGML-using application from displaying data in any way the user desires. Many products provide tools for assigning an appearance to each content object type, and a "WYSIWYG" display for writing and editing.

The Association of American Publishers (1988) has developed a set of SGML "tags" (content object descriptions) for use by member organizations in preparing files for publication. While it is not adequate for scholarly purposes (in part because many needed tags, such as those for poetry, are missing), it does provide a useful example of the design of a document structure. The University of Chicago Press (University of Chicago, 1988) recommends a very similar tag set.

The Text Encoding Initiative is an international committee supported by a wide range of scholarly societies, which is developing guidelines for the encoding of texts for a range of scholarly and commercial purposes.[7] The TEI is a joint European/American project involving scholars all over the world. While not yet near completion, it has already committed to the use of SGML as a basic text description language. TEI guidelines are expected to include standard tags for units commonly of interest, and explicit Document Type Definitions for various kinds of standard documents, as well as definitions of how to extend the guidelines when new content objects need labels.

ADVANTAGES OF OHCO-BASED TEXT PROCESSING

THIS SECTION PRESENTS some of the advantages of text processing systems that use the OHCO model to identify the logic text objects in a document. The advantages are divided into three major categories: composition assistance, production assistance, and facilitation of alternate uses for data. These categories follow the typical life-cycle of a document, not only through publication, but beyond.[8]

ADVANTAGES FOR AUTHORING

THE OHCO MODEL SIMPLIFIES COMPOSITION. Formatting considerations make no claims on the attention of the writer as one composes the text: rather than remembering both (i) the required style conventions and (ii) the formatting commands that are necessary to format the text according to those conventions, one need instead only identify each text element, perhaps by choosing a name from a menu. This allows the author to deal with the document at a level of abstraction appropriate to the role as an *author*, whereas making decisions to embolden or center a title are at a level of abstraction appropriate to a *typographer*.

THE OHCO MODEL SUPPORTS ALTERNATE DOCUMENT VIEWS. The outliners which have recently become standard for use with word processing are merely a minor consequence of an effective model of text. Outline views at various levels of detail can be generated merely by hiding the contents of certain levels of the document hierarchy. A top-level outline view, for example, hides everything but first-level headings. An even more sophisticated selective display of portions of documents is easily expressed in terms of the logical parts of the document.

Unfortunately, typical outliners only consider a limited range of elements; at the lowest level, all is undifferentiated text. This is true, for example, of Microsoft Word's outliner, based on a non-hierarchical text model. Because of this representational limitation, one cannot generally use the outliner to show only poetry quotations, or only block quotes, or only emphatic elements. This pointless and unfortunate limitation follows from treating "headings" as a special case, rather than as a particular set of heirarchical object types, which happens to be frequently useful for generating selective views.

THE OHCO MODEL SUPPORTS TOOLS FOR WRITING. Because an OHCO-style editing program can manipulate a document in terms of realistic components, useful and sophisticated tools can more easily be provided for the author.

Editing software, particularly that which is SGML-based, can be sensitive to document structure because it has a precise description of the content hierarchy. It can provide an outline with all required objects in place, warn the author about any contextually required or prohibited objects, and automatically renumber or otherwise coordinate document components. Softquad Author/Editor$^{\mathrm{TM}}$ (Author/Editor, 1989), for example, shows tags as icons, and prevents certain errors by allowing the user to select tags from menus which list only those appropriate to the current context. Although many editors provide simple versions of these functions, an impoverished model of text limits their effectiveness. For example, Microsoft Word can renumber paragraphs, but due to its non-hierarchical model of text, it cannot distinguish between those that are list items and those that are embedded paragraph divisions within items.

If annotations such as marginalia, footnotes, and bibliographical cross-references are labelled as such, the computer can provide direct connections from texts into bibliographic databases, personal notes, and so on. By recording the structure of these meta-textual components, they too can be used more effectively. FRESS, an early hypertext system, supported some aspects of the OHCO text model, as well as separate annotations, which could then be manipulated with many database-like functions (see Prusky, 1978). BibTeX provides similar functionality for bibliographies.

THE OHCO MODEL FACILITATES COLLABORATIVE WORK. Because the OHCO model describes text in terms which are not specific to a particular formatter or other tool, it makes documents transportable. The same techniques that helps individuals to write and print on their own machine also helps a group of collaborators working on a large project. Since it allows the deferral of formatting choices, and enables authors to have their own set of preferred document views based on a common representation, authors need not conform to each others' styles of work, or waste time making decisions about non-content issues. (The SGML implementation of an OHCO structure also provides the advantage of expressing document structure in plain text, without the hidden binary arcana used by most word processors; this removes many of the common problems of file transfer.)

ADVANTAGES FOR PRODUCTION

GENERIC SPECIFICATION OF FORMATTING: With the OHCO model one need only identify the elements of a document; some or all considerations of final formatting can be deferred. A particular benefit is the ability to simply use generic format definitions or "style sheets" to format a document according to a predefined style. As SGML becomes more widely used, these definitions are becoming available for many publishers' house styles.

CONSISTENCY OF FORMATTING: Because it specifies the appearance of content objects *types* rather than of particular text fragments, an OHCO-based formatter can enforce consistent formatting throughout a document. For example, one can avoid such surprises as finding that one forgot to change the line of one block quote although one changed all the rest. This benefit is available with the style sheet mechanism of current word processors, but is frequently made extremely difficult, since they permit, or even encourage, "exceptions," but give no way to find or remove them. Generally, if an element requires exceptional formatting, it is precisely *because* the element is of a different type, and an appropriate type name should quickly be defined.

ELECTRONIC MANUSCRIPTS: The OHCO model of text allows easy and global adjustment of formatting at any time, since the formatting associated with particular types of content objects is separate from the objects themselves. Once style sheets have been developed for publishers, they can merely be plugged in as needed to conform to any house style. This operation requires no modification to the text *per se*, and thus eliminates the introduction of new errors and the need for repeated proofing.

Similarly, the professional typographer can modify formatting, without having to worry about determining the types and functions of text elements in the original. Consider a philosophical or mathematical text which contains many definitions and many corollaries, tagged as such. If the typographer specifies a layout in which the two classes of objects are typographically distinct, this is trivial to perform even if both had the same appearance in the author's personal style sheet. The same is possible for objects that begin with distinct formatting and end up looking the same. On the other hand, if the distinction was not made initially, neither transformation can be readily accomplished. Someone must re-examine the text and decided for each element whether it is a definition or a corollary, a process which is prone to error.

Output device support can also be handled outside the text file: the result is that the text files themselves are output device independent; only their processing is output device sensitive. Descriptive markup models such as OHCO facilitate generating typeset copy *directly* from an electronic manuscript. The advantages in time, money, and accuracy are obvious.

There are also advantages at the other end of the printing spectrum. Even when a printer has only limited character sets, fonts, and font styles, the text file itself will need no editing before it is printed. The program that prepares the document for printing can be set up to generate the best available appearance for each element. This does not prevent later, more sophisticated uses.

TURNING TEXT INTO A DATABASE

DATA INTEGRITY: Because many of the functions mentioned above allow text to undergo processing — such as reformatting, typesetting, or being ported across text processing systems — without actually editing the source text files, many sources of data corruption can be avoided. Files tagged with descriptive markup are also much easier to port to other processing systems, since they are relatively system- and application-independent. When a text is described as an OHCO, it carries all essential information with it; other types of software can read the mnemonic labels, and understand the data.

INFORMATION RETRIEVAL FUNCTIONS: The OHCO model treats documents and related files as a database of text elements that can be systematically manipulated. This can facilitate not only personal information retrieval functions, such as the generation of alternative views, but also a variety of "value-added" data retrieval functions.

For instance, full-text searches in textbases can specify structural conditions on patterns searched for and text to be retrieved. A scholar might wish to look for philosophical definitions in which the word "sequence" appeared as the *definiens* of a definition, or a philologist might wish to look in a dictionary for all definitions of words containing the prefix "in-" and derived from French.[9] The OHCO model also permits the specification of structural boundaries for proximity searches. This means that a user could look for all chapters whose titles contained both the words "love" and "death." Queries like these, when performed at all, are frequently approximated by specifying general string searches and then hand-culling large amounts of data.

SPECIAL PROCESSING: Many texts include special elements such as formulas in special notations, metrical information, foreign languages, graphical or other non-textual data. This information can be specially tagged and handled by specialized software as needed. For instance, formulas could be manipulated, verified or evaluated; graphics could be displayed.

Since the OHCO model provides a way of representing text that can be decomposed into smaller pieces, it can also be used to integrate a wide variety of different types of data or media into a "compound document." Many current attempts to handle multimedia or compound documents are based on some form of hierarchical content model. Markup that reflects hierarchical document structure can also be used to display and correlate translations easily. For example, CD Word™, a tool for Biblical studies, can synchronize simultaneous displays of the Greek New Testament, English translations, and verse-by-verse commentaries (DeRose, 1989).

The particular complexities of each discipline can be handled intuitively via descriptive markup, because the salient textual units of the discipline determine the tags best used for encoding. Thus, a well-planned tagging scheme facilitates intelligent re-use of data.

USING OHCO TODAY

WILL OHCO-BASED TEXT PROCESSING GAIN GENERAL ACCEPTANCE?

ALTHOUGH STRUCTURED AND CONTENT-ORIENTED TEXT PROCESSING has been recommended by researchers since the 1960s, the microcomputer word-processors of the 1980s have largely ignored the OHCO approach, their designers preferring to emphasize systems that were familiar — that is, similar to using a typewriter — and visually compelling. However, there are now clear signs that OHCO-based text processing will soon be reaching the general text processing markets.

For one thing, users are demanding that the promises of academic computing be kept: they want portability, compatibility, information retrieval, composition assistance tools, publishing from electronic manuscripts, data sharing across applications, and many other things that are not only not available with the most sophisticated microcomputer word processors, but will clearly never emerge in any satisfactory way as new "features."

The most dramatic sign of change is the overwhelming promotion of SGML, the international standard for descriptive markup systems, as a standard for the coding of textual data. Among the organizations endorsing SGML are ISO, ANSI, The American Association of Publishers, the Graphics Communications Association, the European Patent Office, the Commission of European Communities, and others. The SGML application for publishing created by the American Association of Publishers has also been endorsed by the American Library Association, the Library of Congress, the Medical Libraries Association, the Modern Languages Association, the IEEE, OCLC, University Microfilms International (UMI) and other professional and industry organizations.

Most importantly, SGML is a "Federal Information Processing Standard" (FIPS 152) that has been mandated by a number of government offices, including the Department of Defense, for large publishing projects. This regulation is expected to have a substantial impact on the development of the editing and text processing software used by defense contractors developing technical documentation — and this effect will eventually be felt in the larger market of general high-end text processing as well.

Finally, the new microcomputers, with powerful processors and graphical user interfaces, can now support text processors that are OHCO-based but still have the look and feel of WYSIWYG word processors. This was the last obstacle to the creation of popular OHCO-based text processors.

WHAT PRODUCTS ARE THERE NOW?

OHCO-BASED TEXT PROCESSING was pioneered in the commercial world of mainframe batch-style formatters by Scribe and IBM (and Waterloo) Script/GML. These products are still available and provide most of the benefits of OHCO-based text processing. However, the author must implement text structure by using a general editor to enter descriptive markup tags, set off by special delimiters, directly into the text. There are no sophisticated specialized editors with which to facilitate tagging or generate useful formatting on the screen. Consequently, the new WYSIWYG word processors have generally seemed easier to the naive user; the advantages of Scribe and GML are only clear in the case of large or very complicated documents, or documents that are to be typeset.

SoftQuad Author/Editor is an SGML-based editor which runs on the Apple Macintosh$^{\text{TM}}$ and looks to the user much like any Macintosh WYSIWYG word processor. It verifies OHCO structure, provides menus of permissible tags, and has accompanying publishing software which can be used for typesetting. The basic version, however, is not a full-featured word processor, in that it does not have certain niceties such as multiple columns, footnotes, or sophisticated page-layout features. In fact, SoftQuad stresses that Author/Editor itself is a tool for the authors and editors who are developing documents and not for the designers and typesetters who are laying out pages and printing them. The formatting functions it provides are primarily to allow the author to create a natural and visually comfortable editing environment, one where titles look like titles and poetry quotations like poetry quotations. Author/Editor convincingly demonstrates that OHCO-based text processing need not have the old fashioned look and awkward editing of traditional batch formatters such as Scribe and Script/GML.

Software Exoterica sells a variety of SGML tools, including editors, parsers, and so on. These do not as yet provide a familiar word-processor-like interface for convenient authoring, but are more sophisticated in terms of the range of SGML features they handle, and their ability to deal with ill-formed input files. These tools are available for a wide variety of machines.

Interleaf, Context, and many other word processing and desktop publishing vendors are hurrying to support SGML; some see it only as an import/export format, some as a more fundamental design issue. But all are seriously considering SGML, since it has a wide base of support, particularly in large markets such as industries with extremely large documentation requirements.

USING OHCO TOMORROW

MULTIPLE HIERARCHIES

ONE PROBLEM we have not discussed here is the fact that many documents have more than one useful structure. For example, the Bible has at least three disjointed hierarchies which are of use to scholars:

- A reference hierarchy, consisting of testaments, books, chapters, and verses.
- A thematic hierarchy, consisting of pericopes, paragraphs, and sentences.
- For any given edition, a layout hierarchy, consisting of pages, columns, and lines (this is probably the least useful for Biblical texts).

Such multiple structures are hard to represent in any markup system. Though SGML can encode multiple disjoint hierarchies, better representations both at the encoding and the display level still need to be developed. While representing multiple structures in some way is important, making a universal transition to the OHCO model is still a substantial advance and a necessary basis for future improvements.

NETWORKS AND HYPERTEXT

SOME STRUCTURES cannot be fully described even with multiple hierarchies, but require arbitrary network structures. Cross-references and hypertext links are well known examples of such structures. However, even arbitrary cross-references are frequently anchored at portions of the text which are independently motivated OHCO elements, and so an OHCO model of text provides many of the needed "handles" for supporting even these new and sophisticated technologies.

Hierarchical text structures have already been used in the process of constructing hypertexts automatically from pre-existing documents (Frisse, 1987). The divisions of text into small chunks or "nodes" necessitated by the implementation of many hypertext systems is greatly facilitated by the presence of explicitly marked, meaningful divisions in the text. The Perseus Project at Harvard University is actually using SGML as an archival format for their basic texts, which are being included in a multimedia hypertext about Ancient Greece.

VERSION MANAGEMENT

ANOTHER ISSUE that electronic document preparation tools can address is that of managing revisions to a text. This problem appears not only in the form of recording changes made during the authoring process, but also of recording manuscript variants for ancient texts.

SGML appears to provide adequate if perhaps inelegant mechanisms for encoding version information, but little work has so far been carried out in this area.

The OHCO model itself, however, provides information that can greatly ease the process of describing and tracking revisions. Since the OHCO model directly represents such objects as chapters and sections, revision maintenance software has the option to show differences between versions in terms that are meaningful to an author or reader. For instance, a hierarchical model of the text allows the reader to discover that a chapter of a book has been moved bodily from one place to another, as well as to trace any complex changes within the chapter itself. While that information could be extracted by a human looking at a list of differences between the versions, no automatic process could extract that information in the absence of the information describing the document's hierarchical structure.

SUMMARY

"Let me write a nation's data structures, and I care not who writes its code" — W. Richard Ristow.

WE THINK that our point can scarcely be overstated. Text *is* an Ordered Hierarchy of Content Objects; any software application, or any set of computing practices that is based on some other model of text is inadequate for our intellectual and scholarly purposes. Not only will software based on models that ignore content structure prevent us from developing the promised world of "information space," but such applications perpetuate the costly problems of incompatible formats and single-use applications that plague our universities today. No hardware improvements or software "features" can ever make up for this sort of fundamental flaw in design. The functionality we want and need requires that the intellectually salient features of text be reflected in the way that text is organized by our systems — but if those features are not indicated, then no software, however ingenious, can recover them.

What is to be done? First, we must continue to improve our understanding of text structure and pass this information on to our colleagues and universities. Second, we must incorporate what we know about text structure into our computing techniques and practices and encourage others to do so. Finally, we must insist that software developers give us systems that are appropriate to our needs — systems that treat text as a structure of objects and not a "string" of character codes. We, in the university community, should not support format-oriented text processors just as we do not promote faulty methodologies, sloppy research practices, or bad grammer.

NOTES

[1] This paper is largely based on talks given by the authors at Harvard University, Brown University and the College English Association Meeting in Charlotte, N.C. We wish to express our appreciation to the members of Brown's Computing in the Humanities Users' Group, most particularly James S. Coombs, Andrew Gilmartin and Mary McClure, for many helpful discussions and suggestions.

[2] Goldfarb (1981) discusses descriptive markup and describes some of its advantages. See Coombs, Renear and DeRose (1987), for a comprehensive discussion of markup systems and some important extensions, due to Coombs, of Goldfarb's original classification.

[3] There are a few exceptions where order is dictated by publisher's style or is even arbitrary. For example, the order of appendixes, or the arrangement of the many sub-parts of a bibliography item.

[4] This constitutes a limited form of *presentational* markup, along with standard *punctuational* markup extended by traditional conventions. The concept of presentational markup is due to Coombs.

[5] At least, methods familiar to typographers. Recently many of the tasks of production have shifted from publishers to authors. While this has the advantage of giving authors more direct control over the appearance of their work, it also has two major disadvantages: first, the author has less time to spend writing; and second, the typical author is not an expert typographer, and is quite unlikely to produce high quality professional typography.

[6] See ISO (1986). The standard is best read beginning with its appendixes, not with the main text.

[7] The Text Encoding Initiative is sponsored the Association for Computing in the Humanities, the Association for Computational Linguistics and the Association for Literary and Linguistic Computing. It is primarily funded by the NEH, and the European Economic Community. For more information contact the editor of the Project, Dr. C. Michael Sperberg-McQueen, Computer Center, University of Chicago, u35395@uicvm.bitnet.

[8] The following owes much to Coombs; it develops and adapts the discussion of the advantages at descriptive markup in Coombs, Renear and DeRose.

[9] The search software for the new electronic *Oxford English Dictionary* can perform this type of search.

REFERENCES

Association of American Publishers. (1986). *Reference manual for markup of electronic manuscripts.*

Barnard, D.T., Fraser, A. and Logan, G. (1988). Generalized markup for literary text. *Literary and Linguistic Computing.* 13:(11), pp. 26-31.

Barron, D. (1989). Why use SGML? *Electronic Publishing.* 2:(1), pp. 3-24.

Bryan, M. (1988). *The author's guide to SGML.* New York: Addison-Wesley.

Author/Editor. (1989). SoftQuad Author/Editor, version 1.1. Toronto.

Coombs, J., Renear, A. and DeRose, S.J. (1987). Markup systems and the future of scholarly text processing. *Communications of the ACM* 30 (11), 933-947.

DeRose, S.J. (1989). *CDWord tutorial.* Dallas: Dallas Theological Seminary.

Frisse, M. (1989). Searching for information in a hypertext medical handbook. ***Hypertext '87 Proceedings.*** **Chapel Hill, North Carolina, 1987. New York: ACM. 57-66.**

Goldfarb, C.F. (1981). A generalized approach to document markup. ***Proceedings of the ACM SIGPLAN-SIGOA Symposium on Text Manipulation*** **(Portland, Oregon, 1981). pp. 68-73. New York: ACM.**

Toloboff, V. (1986). Trends and standards in document representation. In J.C. Van Vliet (Ed.), ***Text Processing and Document Manipulation.*** **Cambridge: Cambridge University Press. 107-124.**

ISO (1986). ***Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML).*** **ISO 8879.**

Prusky, J. (1978). ***FRESS Resource Manual.*** **Brown University.**

University of Chicago (1988). ***The Chicago Guide to Preparing Electronic Manuscripts.*** **Chicago: University of Chicago Press.**

Weissman, R. (1990). ***Data liberation, or, Goals for a next generation software application architecture.*** **(forthcoming).**

ABOUT THE AUTHORS

Steven J. DeRose received his Ph.D. in Computational Linguistics from Brown University in 1989. He has consulted on a variety of projects in computational linguistics, hypermedia, and related fields. In 1979 he became director of the FRESS hypertext system project, and most recently has served as a technical writer and design consultant for the CD Word biblical hypertext project. He has published papers on descriptive markup, hypertext, natural language processing, artificial intelligence and other topics. He is now the Senior Scientist at praXis, a RI-based firm developing new services and technologies for large-scale electronic books.

David Durand received a Bachelors degree in Computer Science from Brown University in 1983. Since then he has worked for a number of companies in the areas of computer typesetting systems, spelling correction, and operating systems. Currently, he is Manager of Technical Services for Brandeis University Computing Services. He is also serving on the Syntax and Metalanguage Committee of the Text Encoding Intitiative. His research interests include structured text editing interfaces, hypertext systems, and system designs for supporting cooperative work.

Elli Mylonas is completing her Ph.D. in Classics at Brown University. She is currently a Research Associate in the Classics Department at Harvard University, and is the Managing Editor of the Perseus Project. This is a project at Harvard, creating a large multimedia hypertextual database for teaching and researching Classical Greece. She has published and spoken on hypertext, descriptive markup and literary texts and the use of computers in education. She is also on the Text Representation Committee of the Text Encoding Initiative. Author's present address: Department of the Classics, 319 Boylston Hall, Harvard University, Cambridge, MA 02138.

Allen Renear received his Ph.D. in Philosophy from Brown University in 1988. He is on the staff of Computing and Information Services at Brown, where he is responsible for text-related computing, including document management, text databases, SGML and typesetting. He has managed and consulted on many text database and publishing projects, and is involved in a number of information processing standards activities, serving on the advisory board of the TEI, where he represents the American Philosophical Association, and reporting on standards activities, particularly ANSI X3V1 for Xplor, an industry users' group. He has published on descriptive markup and scholarly editing.