Navigation, Access, and Control Using Structured Information

STEVEN J. DeROSE

Abstract: Document representations are gradually moving from format orientation to a more structural orientation, where their internal structure is made available to the machine for more effective processing. This is very much like the move of databases toward more explicit, unambiguous, standardized forms. The data found in documents is quite different from traditional, tabular database records, however, and requires new models. SGML has been at the center of this move because of some very simple characteristics, such as allowing authors to create the particular labels needed for their specific applications. These characteristics led to its use by the Text Encoding Initiative and for Encoded Archival Description. Finding aids pose some uniquely challenging problems for encoding, now addressed by EAD in a way that should make such information far more accessible for the future.

About the author: *Steven J. DeRose has been working with hypermedia document systems starting with the File Retrieval and Editing System (FRESS) hypertext system in 1979. In 1989 he completed his Ph.D. in Computational Linguistics at Brown University and co-founded Electronic Book Technologies. He designed DynaText, the first SGML on-line delivery engine, and other EBT products and served as the SGML expert on the Bentley Library fellowship team that developed EAD. He is now Chief Scientist for Inso, EBT's parent company, and also serves as Adjunct Professor at Brown University. He has written many papers and two books:* Making Hypermedia Work: A User's Guide to HyTime *(with David Durand) and* The SGML FAQ Book.

Introduction

WHEN THIS PAPER WAS originally prepared for presentation at the Berkeley Finding Aid Conference, held on April 4–6, 1995, only a few hardy souls were experimenting with encoding archival finding aids into SGML (Standard Generalized Markup Language[1]). This work used a preliminary Document Type Definition then known as the ''FindAid'' DTD. Since then, archivists have developed a polished tool, Encoded Archival Description (EAD). This SGML application is seeing rapidly growing use for encoding finding aids and making them available in electronic form. The author counts it a privilege to have been invited to help in some of this work (and to have been so kindly received as an immigrant to the field). This paper has been updated to address more recent events such as the completion of the EAD DTD and the appearance of XML, but otherwise it is much as it was in 1995.

What is Structure, Anyway?

Structured information is information that is *analyzed.* Not in the sense that a Sherlock Holmes should peer at it and discern hidden truth (although for some information such as ancient texts, something much like that may happen), but rather in the sense that the information is divided into component parts, which in turn have components, and so on. Only when information has been divided up by such an analysis, and the parts and relationships have been identified, can computers process it in useful ways. The choices made during this analysis are crucial; the most crucial point to be emphasized is that *how you divide up your data does matter.*

There are many models of analysis. Among the most trivial, and in my opinion least useful, is this: ''a document is a list of pages.'' Moving in both directions from this attitude, utility increases. The domain of most concern in the context of EAD is of larger scope, involving the division and organization of recorded knowledge. Without such organization, our libraries and archives would become mere collections, inaccessible, and in the end, unusable. If we move downward in scope, a similar phenomenon occurs: As progressively finer levels of analysis are made conscious, explicit, and accessible, the range of a document's uses increases.

This is what is meant by ''structured documents'' and ''structured information'': information whose parts identify themselves, making them accessible to human and computer processing.

Form vs. Content

One can choose which parts of a document to identify based on many possible conceptual models.[2] Perhaps the first important choice is whether the goal is to represent the *form* of some data or information, or some particular ''meta-'' information *about* the content, or the *content* itself. This choice is fundamental and has radical consequences for what can be done with the resulting information.

[1]International Organization for Standardization, ISO 8879: 1986(E). *Information Processing—Text and Office Information Systems—Standard Generalized Markup Language.*

[2]For further analysis of document representation models and their characteristics, see Steven J. DeRose, David G. Durand, Elli Mylonas, and Allen H. Renear, ''What is Text, Really?'' *Journal of Computing in Higher Education* 1, no. 2 (1990): 3–26.

On paper, form and content are partly intertwined (or as Ted Nelson has said, ''intertwingled''[3]). The typographic conventions of our culture, added to our knowledge of the natural language of documents (sometimes properly linguistic, sometimes graphical or otherwise semiotic), permit us to identify the content parts of books explicitly, with consequent advantages.[4] Computer tools are notoriously bad at identifying content parts when given only form, while being superb at the opposite transformation. For example, it is trivial for a computer to render both book titles and emphasis using italics; this makes for no conflict and requires no artificial intelligence. On the other hand, given two italic portions of a text, computers will fail miserably in distinguishing a book title from emphasis.

Because of this inherent asymmetry, moving into a world of computerized information requires that we undertake the work of making content structures explicit. Without this step, we have not, in fact, moved the information to a new medium; we have merely made an electronic photocopy. Photocopiers are immensely useful, of course, but my point is this: one can do no more with a photocopy than with the original. Certain important gains can be made, such as increasing access while preserving the original from excessive handling, or creating multiple copies for security, or even providing disposable copies for special uses. ''Electronic photocopies'' add the advantage of inexpensive distribution via networks. But, as I stated, once someone obtains a copy, they can do no more with it than with the original.

In our move to the future, we must make it possible to do *more* with our documents. Only by representing the structure of the content, not merely the form of its expression in a prior medium, can we achieve the level of function necessary if we are to manage the exponential growth of information effectively.

Identifying Component Parts and Relationships

Information professionals have names for the many parts that make up document and other information structures. For example, a quick examination of the *Chicago Manual of Style*[5] reveals many such names, since its purpose is to explain how to represent the components of structured content using typographic form.

When creating new information, it is relatively easy to identify the types of content objects. Authors can state authoritatively what their intent is as they place a given content object such as a paragraph, a quotation, a line of poetry, or an axiom. Indeed, they must identify the objects at least implicitly before they can choose a word-processor action to express it. At times authors may be unconscious of these choices, and that is fine—literary works are often held, in retrospect, to be most significant and meaningful at levels their authors may never have consciously considered. Nevertheless, authors' choices of structure are our key source of information about their work, and this holds at all levels, from the phonological and grammatical to the treatment of chapters, indexes, and the like.

When dealing with preexisting information, we do not have the luxury of being its author: we can only do our best to discern structure and meaning from what we have. We

[3]Ted Nelson, *Computer Lib*, rev. ed. (Redmond, Wash.: Tempus Books of Microsoft Press, 1987).

[4]James H. Coombs, Allen H. Renear, and Steven J. DeRose, ''Markup Systems and the Future of Scholarly Text Processing,'' *Communications of the Association for Computing Machinery* 30 (November 1987): 933–47.

[5]University of Chicago Press, *The Chicago Manual of Style*, 13th ed., 1982.

can look for clues to structure in typography (these are often very clear), but we may also wish to find structures that are completely implicit or that are obscured by neutralization. For example, when the *Oxford English Dictionary* was converted into a structured electronic document, researchers found roughly twenty distinct uses for italics;[6] only by the painstaking task of teasing etymologies apart from Latin cognates apart from literary examples, and so on, was the result made truly useful. At a subtler level, one may wish to explicate structures that are hypotheses: a literary critic may claim that some passage constitutes an allusion to *Paradise Lost*. The validity of such a claim normally remains debatable, but explicit structure is a way to express the claim itself.

The key innovation required to move forward is that we must choose truly useful structures and make them explicit. The structure will be there anyway, but using it must remain a purely manual task unless it is made explicit through document markup.

Why Do We Need Structure?

Structure is Really There

Structure is in our documents. We cannot avoid it, though we can choose what kind to use in any situation. Authors think in terms of linguistic discourse and other structures while writing, though much of this activity becomes automatic with practice. We also use structure constantly in navigating the information we have. Finding aids have a great deal of structure, which is created by archivists through careful design.

We often make use of structure unconsciously. Examine any document and structure leaps off the page: lists, figures, footnotes and the like are pervasive. As documents grow larger, explicit structure-aware tools start to appear: indexes reflect the thematic or topical structure, tables of contents reflect the broad-stroke discourse or organizational structure, and bibliographies reveal something of the referential or link structure. In reference works, such as those of particular interest in the context of archival finding aids, structure is even more important. Without carefully designed subject categories, levels of organization and description, and other structural techniques, navigation in large information spaces bogs down.

Structure Provides a Way to Name Things

Raising the component parts of information to the level of explicit representation often leads to giving them names. As Ursula LeGuin reminds us, "the name is the thing, and the true name is the true thing. To speak the name is to control the thing."[7] Nowhere is this truer than in the realm of information.

Navigation requires naming, as does access whether by database, catalog, finding aid, or hypertext link. Choosing the right names for information units is perhaps the most crucial issue facing the electronic document community today. We have spoken already of type-names, which identify what manner of thing a thing is. Let us now turn to instance-names, which identify specific individuals: not X is *a* book or quotation or word or link, but X is *that* book, or *that* quotation, or *that* word, or *that* link.

[6]Frank W. Tompa, "What is (tagged) text?" *Dictionaries in the Electronic Age: Proceedings of the Fifth Annual Conference of the UW Centre for the New Oxford English Dictionary* (Waterloo, Ont.: University of Waterloo Centre for the New OED, 1989), 81–93.

[7]Ursula LeGuin, "The Rule of Names," in *A Treasury of Fantasy: Heroic Adventures in Imaginary Lands* (New York: Avenal Books, 1981), 495–504.

Imagine for a moment that we lacked such names for information units: what if there were no chapter, section, or even page divisions for authors to reference? This is almost inconceivable at the level of whole documents: a book without a title will be given one or will die a quiet death. But what of internal components? Ancient texts lacked internal names, and the important works have been forced to acquire them. One can hardly find a modern Bible printed without chapter and verse divisions, and the same is true for scholarly editions of most classical works. Manuscripts often lack such internal cues, making the texts before us that much more complex.

For recent works we resort to page numbers for cross-reference; for example, "see page 37 of Smith (1995)." This is possible because the number of copies whose pagination matches is very high; many books never achieve a second edition, or even a second printing. But for those that do, the use of page numbers poses a problem that brings us back to structure: *page numbers break.* This is obvious, but easily forgotten:

- A large-print edition cannot be published without either making the pages physically huge and unwieldy or making the page numbers inconsistent and therefore useless for edition-independent reference. This is inherent with pre-formatted data, and is easily seen in most word processors: one cannot narrow the window without clipping off the end of every line.
- Even a tiny change to the content may break all later page numbers, and such effects are cumulative.

Why do these things happen? It is simply because pages are not structural units in literature. They are certainly structural units in the far different domain of typography, but typography is not our context of interest. A book is "the same" if reprinted from quarto to octavo and from Garamond 24 to Times 12 in all but a few senses.

Precisely the same issue affects reference tools such as finding aids. What if the only names for things were chosen from a space that itself had little structure? For example, imagine a library organized and accessed solely by ISBN or acquisition number, or a finding aid lacking levels of organization. While the presence of names would enable minimal access, a radical loss in functionality would be inevitable.

Structure vs. the Alternatives

The careful choice and use of structural elements, together with the careful assignment of systematic element names, provide the tools required to navigate the vast information-spaces that are just around the corner.

Many proposals have been made to utilize only the notion of pages in the electronic world. The most naive form may be "Just scan everything in LC and drop it on the net." A few years ago one heard the same theory, but suggesting optical disk jukeboxes, and before that, microfilm. Such approaches, even ignoring obvious feasibility problems, would not truly achieve the benefits expected of a new medium. Unstructured data forms such as the bitmap are merely new kinds of papyrus on which to make copies: highly useful but purely a quantitative, incremental change. This path can never lead to the new world of navigable, accessible information space that we hope to attain. It carries over most weaknesses of the paper medium, while failing to retain paper's compensating strengths.

This is because a scanned image does not contain explicit structural information that can be used to support computer processing that could add value. For example, one could build an "electronic catalog" by simply scanning three by five cards and then saving the bitmaps without performing optical character recognition. Such a catalog could be "on-line'' and would have the advantage of being easily copied, backed up, and transported. But imagine using it!

The next step up from mere pictures of information was once very popular: the ''plain ASCII text file'' sings the Siren song of portability and is indeed more amenable to machine processing than a bitmapped page. It can be word-searched or mailed around, and nearly any software can at least display it. But to put the seeming advantages into perspective, one must also consider the costs. For example, many important information structures cannot be represented in ''plain ASCII,'' including:

- *Foreign languages*: Any characters not in the very restricted set used by English, such as French accented vowels, not to mention the deeper difficulties of Greek, Hebrew, and Japanese.
- *Footnotes*: Where do you put them, in-line, or at the bottom of the page, or at the end of the book? How are they identified as footnotes in the first place?
- *Running headers* and any other constructs where information recurs, or appears without being part of the ''actual'' text, or appears out of order.
- *Graphics*, charts, and other nontextual information.

Beyond these obvious limitations lies a subtler problem: ''plain ASCII'' files often use idiosyncratic conventions to represent information about structure. For example, block quotes may be indented by adding spaces before each line, or titles may be indicated by adding enough spaces to ''center'' them. Such conventions can add potential for more useful functionality, but such a file is no longer ''plain ASCII.'' Some of the characters are no longer just characters: they have become markup, or ''metadata,'' giving information *about* the text. The main difference between such conventions and true markup is that the conventions are inconsistent and undocumented. Many interesting and desirable electronic texts are freely available in ''plain ASCII,'' but a closer look sometimes reveals that these texts are not all they claim to be.

- *Gaps* are often present, such as a missing chapter (possibly due to copyright problems, but that is another set of issues).
- *The source edition* often is not identified (after all, it's not ''part of the text''). Identifying the source text can be a formidable task in documenting archival materials, and so it is essential that provenance information be retained (and marked up) when it *does* exist.
- *Footnotes*, graphics, accented characters, sidebars, and other display elements are often missing or misplaced.
- *Page and other references* are lost with no provision of a substitute, so important tools such as indexes, tables of contents, and cross-references are either deleted or useless.
- *Typographic nuances* such as emphasis are lost, even though they may be crucial to understanding the text. For example, I recently read a magazine article on world hunger which included the sentence ''World hunger is not *a* problem'' with the ''a'' in italics. The point, of course, was that hunger is a complex of many problems, from the biological to the political. Keep the ''a'' and delete the italics as ''plain ASCII'' must do, and the meaning changes radically: ''World hunger is not a problem.''
- *The lowly spacebar* attempts to carry the full load of representation, leading to insoluble ambiguity such as a title that happens to be long enough that ''centering'' it indents it exactly as if it were a paragraph. How do you tell what you've got?

Pity the scholar who analyzes such a text, or the cataloger who tries to identify it. *The names we need are missing*. In LeGuin's terms, we do not know the true name and so cannot control the thing. And if, as in her story, we should magically learn the true name, we find to our pain that the thing we name is not what we thought—not an unassuming local wizard, but a dragon in disguise.

Structure Provides Handles for Searching

My final point about the need for structure is that structure facilitates searching. Only if component parts are explicitly identified can we search for information *in some particular part*. This is why a database of personnel records is better than a list typed into a word processor. In a database, one can search for ''Jones'' as a name but not a street, or for ''401'' as an area code but not a street number. Likewise, no one could sell a personnel database where a search for numbers ''>10'' could not specify whether this refers to a ''salary'' or ''month of hire.'' And in my favorite example from one on-line library catalog, it is painful to search for the journal titled simply *Linguistics* if you cannot avoid all the materials indexed under ''Linguistics'' as a subject. Such cases are so obvious that we may hardly think of them as ''structure,'' but as full-text documents are put on-line, the same issues and tradeoffs apply. If we do not represent structure within documents, we cannot do the kinds of processing we increasingly want to do.

Many finding aids occupy a typological middle ground between databases at one end (especially simple flat-form databases, or the more complex and heterogeneous databases such as MARC) and typical documents at the other. This makes finding aids even more complex and in need of careful design than many other data structures. This continuum from simple flat-form databases to highly structured document bases raises the issue of what kinds of structure to represent. As we move from catalogs and abstracts toward finding aids and eventually to full content, correlating the levels of information and using levels and structure to increase ease of use will continue to grow in importance.

What Kinds of Structure Are Needed?

Basic Kinds of Data

I would like to suggest a few basic kinds of structured information, ranging from *forms* at one extreme to *documents* at the other, and then to argue that certain reference materials ranging from MARC records to finding aids fall along the continuum in between. I do not think that finding aids fall cleanly into either extreme: I think that because of their intermediate nature, they have both advantages and difficulties not present at either extreme. First, let us consider forms, such as those we all fill out from time to time on a sheet of paper with little boxes or in a relational database (RDB). Form or tabular data has these central characteristics:

1. *Many instances* of the same group of information items—that is, many copies of the same form. Particular instances of a form may have some items left blank, but if there are many such items we suspect a bad form, because not all instances are comparable. Likewise, explanatory notes about such variations are considered signs of a bad form.
2. *Order is not important* to the meaning, although the information on a form is inevitably presented in some order. The order of the instances of a form is also irrelevant. For example, the order in which employment applications appear in a paper

or computer file is generally irrelevant. Perhaps "who filled out the form first" matters, but that is quite a different issue.

3. A form's *context is not part of its meaning*. More concretely, taking one instance of a form out from among its fellows does not change its meaning. This is crucial to the way in which form databases work: a report or the result of a search is a list of form instances isolated from their fellows; each makes full sense independently of the others.
4. Forms also involve a subtle notion of the *identity of information*. If two forms are filled out exactly the same, they are indistinguishable for all processing purposes: they are *the same*. This is why companies assign customer and order numbers, and why it is so troublesome when the same number is accidentally assigned twice.
5. Items on a form have *little hierarchy*, which is to say that there tend to be few item/subitem relationships. A person may have both home and business addresses, each with several parts, and it would be wrong to mix the street address of one's home with the zip code of one's business, but such examples are few and are provided explicitly. Forms cannot have unbounded repetitions of structured subparts.

Documents, at the other extreme, have quite a different pattern regarding the same central characteristics:

1. *Few instances* of a given sequence of pieces of information; for example, it is pure coincidence if two books have the same number of chapters and sections. It is odd to think of a book or article leaving certain structural elements blank, such as chapter one—even a nonstructural unit such as a page is amusing if it is "intentionally left blank" on paper, and is absurd on-line. Likewise, "explanatory notes" such as footnotes, sidebars, and digressions are the norm in documents.
2. Unlike information stored as forms, the *serial order* of most information in a document matters to the meaning. It matters greatly which paragraph comes first,[8] and this poses a deep performance problem in the relational database model so useful for other kinds of data. An RDB would typically store each paragraph (or section, or other unit of text) as a record, and records are by definition unordered. To produce the correct order, serial numbers must be added. The RDB must select records with serial numbers in a certain range (likely a slow operation) and then *sort* them. This is wasted effort if, as with documents, a single basic order is almost always needed, yet must be reconstructed over and over. A database model that *preserves* order saves all this work.
3. *Context* matters regarding information in documents. While for form data, taking one instance out from among its fellows does not change its meaning, the opposite is clearly true for document data. This too is crucial to the way in which document bases work: The result of a search is not a list of small components isolated from their fellows, but a component *in its context*. Some time ago a document query

[8]Information order is central to hypertext theory, though Ted Nelson's definition of hypertext as "non-sequential writing" seems to contradict this. I take Nelson's definition to mean writing that is not *strictly* sequential: not locked into a single sequence as imposed by the paper or film medium. Authors instead give their readers many choices—this is precisely where poor hypertext systems and documents most frequently fail. Yet even the most labyrinthine hypertext is highly sequential. Even Faulkner must make some passages prerequisite to others. So while hypertext goes radically beyond the idea of a *single* sequence or even a small number, it does not overcome time, language, and cognition. This is why authors carefully craft hypertext links, rather than merely having a computer draw and quarter the texts.

language was proposed that lacked this key feature. In it, a query for all occurrences of the word "sower" would get sower, sower, sower, etc. What one must have is the list of where "sower" occurs, so as to navigate to those places and examine the context.

4. Unlike forms, two *identical objects* in a document are not necessarily the same. It is possible for a word, a sentence, or even a paragraph to be repeated in a document, and if this happens, the repetition matters. These repeated instances do not duplicate each other.
5. Finally, while forms have little *hierarchy*, layers upon layers of substructure are a hallmark of documents. This characteristic is even more pronounced in finding aids than in most other documents.

Forms and documents differ radically on all these fundamental axes. My conclusion is that different tools and methods must be applied in the two domains, and indeed the history of document processing systems has (with some digressions) followed this course.

Where do finding aids fit in? I believe they share some characteristics of both categories, and this may make them particularly complex. A finding aid must include a great deal of information about content, since that is what one is trying to find. Some meta-information in finding aids can be reduced to something resembling forms; in one sense, a finding aid is similar to a MARC record: a large though typically sparse list of fields. But there is more going on. MARC fields do have interdependencies, they do have levels (a colleague working on the John Carter Brown Library's bibliography of European Americans ended up dividing author names into something like twenty subcomponents and three or four levels). But finding aids have a much more detailed and complex hierarchy, and so presumably must go even further.

A finding aid must provide access based not only on demographic information—author, title, subjects, added entries, and a host of other fields—in addition, it must make use of characteristics of the content itself. What is this collection or group of records about? What school of thought from the discipline does it represent? What does it relate to in other disciplines?

There is especial benefit in being able to determine relationships as yet unnoticed or unremarked. Markup that identifies relevant content and structure facilitates such a discovery process by making explicit many of the basic facts upon which conclusions about relationships are based. One approach to this is the preparation of abstracts, and this has proven very useful. Another is the application of statistical methods to vocabularies and word frequencies, now well understood. But the ultimate answer, I believe, comes from making whole documents available with as much structure as possible explicitly represented. This is the true information, labeled by true names, from which abstracts and statistics come.

Particulars of Document Structure

How then do we represent useful structure? Many parts are obvious, and, within the hard constraints of time and budgets, we should represent as many of them as possible.

First, almost all documents include various *generic component parts*: PARAGRAPH, LIST-ITEM, QUOTE, TITLE, EMPHASIS, FOREIGN, IMAGE, and the like. Even rudimentary software can help locate such parts, because they map almost one-to-one to word processor or scanner objects. Reasonably skilled yet not scholarly workers can identify them quite reliably, even when software cannot.

Second, there are various *generic aggregates*: BOOK, CHAPTER, SECTION, FRONT-MATTER, LIST, TABLE. For historical reasons, however, typical software gives no help with these. We have all suffered from word processors not knowing what a ''list'' is and failing to number items correctly, not keeping numbers up to date, or forcing us to reselect each list each time we add or delete an item, all because the software can't remember any unit larger than a single paragraph. We suffer the same pain when we want to move, delete, or otherwise deal with sections and chapters. Add-on outliners help, but most word processors lack structural knowledge and instead use heuristics (such as ''Find the next paragraph of type HEADING-2, and assume everything between is the current SECTION'') that are both slow and unreliable.

Third, each genre, from poetry to manuals to finding aids, requires *specialized objects*: STANZA, REPAIR-PROCEDURE, AXIOM, PART-NUMBER, CATALOG-CODE. Identifying the right ones for finding aids is a crucial step, requiring ongoing research. A closed set of elements cannot be established once and for all, just as the list of defined subject headings for literature cannot be defined once and for all. The Berkeley Finding Aid Project undertook this task with zeal, and the EAD tag set that has been developed based on a large corpus of finding aids promises many advances in the portability and accessibility of such information.

Fourth, there is ever-increasing need for *access tools*: these range from the ubiquitous footnote and sidebar to cross-references, bibliographies, and the like. Use of paper documents necessitated other navigation tools as well, such as indexes and tables of contents, and of course these components should be represented. Most of these access tools can be expressed on-line as one or another type of hypertext link: any such reference should be linked to its source, as should any quotation. As referenced documents change through critical editing or a rewrite by a living author, the user may wish quotations to be dynamically updated. HTML and the Web have made one very basic type of hypertext link and one kind of name (the URL) ubiquitous, yet there remain many other and more powerful types of linking and locating, now coming into mainstream use through efforts such as the Extensible Linking Language (XLL), part of the Web Consortium's XML effort.[9]

Some phenomen a evident in printed texts are not structural units that need to be identified for most purposes. Line breaks, discretionary hyphens, font and other typographic choices, and the like usually are not structural except insofar as they may serve to communicate some other structures. This is precisely why electronic delivery methods that closely mimic paper have problems: they transfer ephemera and accidents of typesetting with as much primacy as they accord the very words and structure of a document, even though such accidents usually are liabilities rather than advantages in the electronic medium. The most obvious example is font size, which can safely be far smaller in print than on-line, yet must be slavishly followed in any page-fidelity-driven approach.

How to Decide What's Structure?

When planning an encoding project, two primary questions are ''What structures are of interest?'' and ''Which are to be encoded?'' *How* structures are encoded is important, but strictly less so than the *fact* that they are encoded. Any encoding project faces economic as well as intellectual decisions, and I will not address how to decide which things *not* to encode when finances are limited; this depends on the goals and usage scenarios envisioned

[9]For more information on XML see <http://www.w3.org/TR/WD-xml>.

for the data. My normal advice on the subject is that within the constraints of budget, encode anything likely to be of independent use later. Here are a few specific diagnostic questions to ask about a component under consideration for structural markup:

- Does it survive reformatting the document?
- Is it useful for multiple purposes?
- Would an author or reader have a name for it?
- Might someone want to search for it specifically, or constrain text searches to it alone?
- Does it surround, fill, or associate with other particular units?

This list clearly is biased toward conceptual units at the expense of the merely typographic, in keeping with the state of the art in document encoding. Deciding which structures to encode is, fortunately, not an embryonic field, nor is the use of particular structural representations such as SGML and its offspring XML. There is much good work to build upon, such as that of the Text Encoding Initiative (TEI),[10] Encoded Archival Description, and many other projects in varied domains.

How Does SGML Fit the Bill?

SGML is the best choice for encoding conceptual structural units in documents. It has two crucial advantages: First, SGML imposes no fixed set of component types. You can define the structures as appropriate for the task at hand. At the Center for Electronic Texts in the Humanities (CETH) *Workshop on Documenting Electronic Texts* held in 1994, one of the speakers expressed some doubt as to whether SGML was flexible enough to provide a complete equivalent to MARC (that is, an alternative representation of all the same data). By the next coffee break, three of the SGML experts present had drafted DTDs (hardly polished of course, but sufficient proofs of concept).[11]

Second, SGML is a public, nonproprietary standard that will not change with each new release of a company's software. Software vendors conform to SGML, rather than SGML and an individual's data having to conform to particular software vendors. This is what justifies confidence that SGML data will survive for the long term, beyond any current software used to process it.

Third, SGML provides a very direct representation of data. As with HTML, other SGML can be read even with *no* specialized software; there is no binary hash, interpretable only by intermediary software. This contributes greatly to data longevity, even though it slightly complicates SGML's means of representing nonhierarchical structures such as links. So are there any downsides to SGML? Only a few. To some it may seem problematic that SGML requires more thought about the data. OCR and proofreading are no longer the end of the data integrity story, but the work must continue into making sometimes difficult decisions about the nature of your data. I consider this an upside: it does require extra effort, but the effort generally pays off.

[10]C. Michael Sperberg-McQueen and Lou Burnard, eds., *Guidelines for Electronic Text Encoding and Interchange* (Chicago, Oxford: Text Encoding Initiative, 1994). Also available on-line from ftp://ftp-tei.uic.edu/pub/tei and many other places, most of which are pointed to from http://www.sil.org/sgml/acadapps.html#tei (part of Robin Cover's extensive SGML information guide).

[11]Lisa R. Horowitz, *CETH Workshop on Documenting Electronic Texts, May 16–18*, 1994, *Radisson Hotel, Somerset, N.J.*, Technical Report #2 (New Brunswick, N.J.: Center for Electronic Texts in the Humanities, 1994).

The main downside to SGML is that it provides too many options: alternative syntax, abbreviation conventions, and the like. Few people bother to learn them all. Fortunately, such options are just that—options. Many do not add functionality or capability, merely alternative methods, and so any project can simply choose to avoid them. Thus most SGML experts have adopted what has come to be called a ''monastic'' approach—''just say no'' to any features you don't need.

This is precisely the strategy that the Extensible Markup Language (XML) uses. XML has all the extensibility and representative power of SGML, but without the syntactic complexity. For example, implementing an XML parser requires roughly a week's work, rather than a year's; this allows the focus to shift from technical implementation details to data analysis, which has more value in the long term. The other ''X*L'' standards, XLL and XSL, achieve similar grand simplifications of existing standards for hypermedia linking and for stylesheets.

Summary

Great progress has been made in the last decade in shifting electronic information from purely format-driven forms to more processable, flexible, structure-driven forms. Processable finding aids are one of the next logical steps as archivists strive to progress from information about the *form* of documents toward treating documents themselves as sophisticated information objects. At each stage, what the computer can do with data depends most importantly on the model applied to the data. A simple facsimile of a manuscript or other object is useful but it does not enable qualitatively new processing, just as a microfilm copy of a card catalog is useful but not revolutionary.

In designing new models for electronic data, it is important to consider where traditional models such as the relational database do and do not fit. In examining several basic properties of relational data versus documents in general, it becomes clear that the ''fit'' is questionable. Newer technologies are needed, and new design questions need to be researched and solved. It is also increasingly important to create, manage, search, and maintain ''metadata'' (information about other data). Metadata comes in many forms, from PICS ratings of Web page appropriateness, to cataloging information, to critical reviews and commentaries. Standards for representing such metadata for Internet use are a very recent and interesting development.

SGML provides a generic way of representing document structure models, and of representing documents and other data given those models. Because SGML is a formal standard and has achieved widespread and diverse use, it is a safe long-term vessel for important data. As with many standards, a streamlined approach to SGML, such as XML, enhances portability, durability, and interoperability by attaining the freedom of simplicity.