Implementing EAD in the Yale University Library

NICOLE L. BOUCHÉ

Abstract: In 1995 the Yale University Library agreed to serve as an "early implementer" site for the FindAid DTD, now EAD. Three Yale manuscript repositories—the Beinecke Rare Book and Manuscript Library, Special Collections in the Divinity School Library, and the Manuscripts and Archives Department of Sterling Memorial Library, agreed to participate in this test phase by creating an integrated database of SGML-encoded finding aids, accessible via the World Wide Web. This case study describes the basic configuration of the Yale finding aid site, reviews the project's history and accomplishments since its inception in 1995, discusses the methods employed by the participating units to mark up legacy files and to integrate EAD into their routines for creating new finding aids, and suggests issues to be considered by others contemplating an SGML/EAD encoding project of their own.

About the author: *Nicole Bouché is Head of the Manuscripts Unit at the Beinecke Rare Book and Manuscript Library at Yale University. She directs the Beinecke's implementation of EAD and is actively engaged in the further development of the Yale Finding Aid Project site and related digital library initiatives at Beinecke. From 1987 to 1993, she was Assistant Head of the Manuscripts Division of the Bancroft Library at the University of California at Berkeley. This is a somewhat revised version of a paper given on 29 August 1997 at the annual meeting of the Society of American Archivists held in Chicago.*

Introduction

THE YALE FINDING AID PROJECT is a joint venture of Yale manuscript repositories and the Library Systems Office to create an integrated database of finding aids, accessible via the World Wide Web. The Yale website[1] contains finding aids from three Yale library units: The Beinecke Rare Book and Manuscript Library, Special Collections in the Divinity School Library, and the Manuscripts and Archives Department of Sterling Memorial Library. At the present time, the site contains over 350 SGML ''instances'' (i.e., finding aids) encoded using the beta version of Encoded Archival Description (EAD). HTML versions of the SGML files also are available. Additional finding aids are added to the site as they are completed, and in fall 1997 the Beinecke Library loaded almost three hundred preliminary finding aids in pre-formatted HTML, derived from plain ASCII text files. Files in other non-SGML ''native'' formats (e.g., WordPerfect, Microsoft Word) may be added at the discretion of the contributing repositories.

Although only three Yale repositories undertook initial development of the finding aid site, now that a core group of Yale ''early implementers'' has acquired sufficient expertise in implementing EAD, building the finding aids database and maintaining the server, we anticipate assisting colleagues at Yale in adding their finding aids to the site. Eventually, members of the Yale community, as well as researchers elsewhere, will have unprecedented access to an integrated source file of finding aids describing an extraordinary range of primary source materials scattered throughout Yale's various research and departmental libraries and museums.

A project of this scope has many facets; this case study will touch on only a few. First, I will describe the basic configuration of the finding aid site and review the project's history and accomplishments since its inception in 1995. I will then discuss the methods employed by the three participating units to mark up legacy files and to integrate EAD into their routines for creating new finding aids. Finally, I will close with a few recommendations, based on our experience over the past two years, for those who are considering an SGML/EAD encoding project of their own.

The Setup

Yale's EAD-encoded finding aids reside on an OpenText (OT) server running OT's search engine software LiveLinks. The finding aids are cross-indexed: one can search across the holdings of all three contributors or just the finding aids of a single unit. A finding aid can be accessed in three ways: via links on the repository's website, by directly searching the finding aid server, or by clicking on the linked 856 field in the corresponding MARC record in the Webpac version of Yale's on-line catalog, Orbis.[2]

The finding aids website supports both simple and more complex key word searches: standard Boolean operators (AND, OR, and NOT) and proximity qualifiers (WITHIN and FOLLOWED BY) are incorporated into the search forms. Searching the SGML-encoded files by specific EAD tag regions also is possible. At present, EAD tag regions defined and indexed for searching purposes across all SGML-encoded finding aids are Entire Document, Introductory Material, and Container List. A limited number of other tag-qualified search op-

[1] <http:\\webtext.library.yale.edu>.
[2] <http:\\webpac.library.yale.edu>.

tions is offered at the repository level, as indicated in pull-down menus on a unit's SGML-Encoded Finding Aids search form.

Although many other indexing and search configurations are possible, implementing them would greatly complicate file markup as well as the routines for loading and indexing the files on the server. For this reason, the EAD Implementation Group settled on this initial set of indexing and search options. We recognize that, in time, other configurations may be preferable, in which case the indexing structures, search forms, and results screens can be adjusted or redesigned accordingly.

Once a finding aid has been identified on the OpenText server, it must be brought up on the individual's PC by means of an appropriate Web browser (e.g., SoftQuad's Panorama viewer for the SGML-encoded files, or Netscape for the simple ASCII and HTML files). Files in other "native" formats such as Microsoft Word or WordPerfect similarly require the operator to select an appropriate application to display the file. As with other Web queries, a search for a specific term or phrase used initially to identify finding aids on the OpenText server must be reinitiated within the finding aid itself to locate specific occurrences of the search term(s).

Project History to Date

In the early days of the Berkeley Finding Aid Project (BFAP), from which EAD evolved, the Manuscripts and Archives Department in Sterling Library was one of several repositories in the United States that supplied copies of finding aids to the Berkeley group so they could study the feasibility of developing a national standard for finding aid encoding and Internet delivery. The Yale library as a whole, however, did not become involved in the project until after the Berkeley Finding Aid Conference, held in April 1995. Richard Szary, Head of Manuscripts and Archives, and I, representing the Beinecke Library, attended the conference, and we were much intrigued by the progress that the Berkeley team had made. After reporting back on their findings to our colleagues at Yale, we received enthusiastic support for the Yale library to become one of the early implementers of the new encoding standard. It was fortunate that the BFAP conference and the subsequent release of the FindAid Document Type Definition (DTD) for wider testing and evaluation by the archival community coincided with a growing interest in the Yale library in installing an SGML-capable server to enable the library to expand the repertoire of online reference tools and services that it offers.

Consequently, within a few weeks of the Berkeley conference, the Beinecke Library had committed funds to purchase an SGML server and the necessary start-up software, and a hardware/software selection task force had been appointed. Three Yale library units volunteered to be the Yale "early implementers," and essential technical expertise was assigned to the project from the library systems office. By fall, the technical task force had selected a start-up set of hardware and software that included an OpenText server, SoftQuad's suite of SGML authoring, publishing, and Web-browsing software, and an SGML-enabled version of WordPerfect. With all the basic tools in hand, a formal EAD Implementation Group was appointed in February 1996. The team included representatives from the three participating repositories, the library systems office, and two additional staff members representing the public services department in Sterling Library and the Social Science Library. (The latter was interested in other forms of SGML implementation.) Within a matter of weeks, however, as the mechanics of simply getting an EAD site up

and running became clearer, the task force had narrowed itself down to a core group of EAD implementers.

Although the archivist/librarians on the implementation team were well versed both in finding aids and in the research and reference patterns for their departments, our knowledge of SGML was virtually nonexistent, and our experience with Web publishing was limited at best. Although all of the participants were producing finding aids in machine-readable form, only Beinecke's guides were available to researchers on-line, both locally in a stand-alone database and on the Internet via the Yale Gopher. As a general rule, project staff had little experience searching finding aids in an on-line environment or maintaining them on a server. With one key exception (Beinecke), no one had any real experience in programming or writing macros for data processing: it became clear fairly early that the Divinity Library and Manuscripts and Archives Department would need to recruit additional technical assistance from outside their units, while Beinecke would have to draw on additional in-house expertise.

As one might expect, the systems personnel assigned to the project knew little about archival practice, patterns of research in archives, or the creation and use of finding aids; their contributions came in the form of an invaluable range of experience in library automation. Beyond the obvious assets of programming and technical expertise, the systems staff had experience in the routine maintenance and ongoing development of the library's OPAC and in the installation and maintenance of other library servers. They also had recently installed the first Web version of Yale's on-line catalog.

From February through summer 1996, the EAD Implementation Group met weekly to work out the mechanics of everything from basic SGML encoding to HTML search form design for the website. The process of mastering and implementing EAD was very similar to the early days of learning MARC AMC, before there were workshops or full-blown documentation with examples. When we started, the tag library wasn't even available. (It came later, with the alpha release of the EAD DTD.) All we had to work with were the FindAid DTD (which, at first glance, was utterly unintelligible), later the alpha EAD, and a handful of sample finding aids, including a few of our own, which the Berkeley team had marked up the previous April.

At the repository level, task force members reviewed current finding aid practices and experimented with markup by hand. (Virtually from the outset, we had determined that markup would have to be automated, so this initial manual work was simply to become accustomed to SGML and to EAD, as well as to devise our tagging protocols.) At weekly task force meetings, the three teams compared results and revised and clarified tagging practice to achieve consistent tag interpretation, if not uniform finding aid style. We eventually settled on a core group of 44 tags to be used for Yale finding aids out of a beta EAD tag library of approximately 135 tags. The more technically proficient members of the task force also began to experiment with the various authoring and publishing software packages that had been acquired as part of our start-up package. Eventually they mastered the SGML publishing routines of Panorama and developed preliminary style sheets and navigators for each repository.

Meanwhile, systems personnel worked to install and configure the server and search engine. An initial group of test files using the EAD alpha version was loaded in early summer 1996. Finally, a year into the project, we began to get a sense of how all the pieces might actually fit together as an integrated, Web-based source file of Yale finding aids. Having acquired a working knowledge of both EAD and of Panorama, and with our basic tag library and tagging sequences mapped out, the teams in each unit turned their

attention to more systematic encoding methods. We spent the summer of 1996 developing more efficient automated tools to mark up legacy files and to introduce SGML-encoding into our respective routines for producing new finding aids. As with our early experience with tagging, there was considerable discussion and exchange of ideas about strategies for writing macros to expedite markup as each unit worked out methods and tools appropriate to its particular circumstances.

At this point in the project, however, the experience of the three repositories began to diverge because of fundamental differences in both established routines for creating new finding aids and the scope and character of our legacy files. At Beinecke, conversion of legacy files surged ahead, and mechanisms were readily put in place to integrate SGML encoding into routines for producing new finding aids. Work in the Divinity Library and Manuscripts and Archives Department proceeded more slowly, especially in Manuscripts and Archives, which faced the largest and most complex backlog of legacy files (estimated at eighteen hundred finding aids.)

In August 1996 the Yale Finding Aid Project website, containing fifty EAD instances, was announced in "test" mode to the EAD listserv.[3] Between that time and May 1997, when the site was announced to the archival community at large, file conversion continued in each unit to the extent possible, given staff availability and the inherent difficulties of file conversion. Overall, the project weathered a string of technical modifications and upgrades that affected both the OpenText server and the various components of SoftQuad's suite of SGML publishing and browsing software. Relatively late in the game we decided to provide HTML versions of EAD instances for the benefit of those unable to access the SGML-encoded files, and so a parallel set of files was created and added to the server.

When the Yale Finding Aid Project site finally went public in May 1997, it contained 323 EAD instances: 260 for Beinecke, 46 for Divinity, and 17 for Manuscripts and Archives. Development work on the overall site is ongoing, and we continue to build the database with new and updated files.

Some Key Early Decisions

Besides generally accepting the logic of the EAD DTD and the content of the beta EAD tag library, the Yale EAD Implementation Group also made some strategic decisions in the early days of the project that influenced our selection of hardware and software, the character of our EAD instances, the configuration of files and indices on the OpenText server, and the formatting of search and results screens. These decisions, once made, enabled us to proceed fairly expeditiously to our greatest challenge—file conversion—without any of the contributors having to significantly modify either the format of their finding aids or their workflow to accommodate our EAD implementation.

Our selection of the initial suite of hardware and software (within what was admittedly a rather small universe of commercially available options) was guided to a great extent by two key considerations. First, the Yale library wanted to install a powerful server

[3]The EAD forum is an electronic mail posting service created to facilitate the exchange of information about the SGML DTD being developed for archival finding aids. Developers and implementers of the EAD DTD are welcome to join the forum and share their ideas and experiences with using SGML for finding aids. To subscribe to the EAD forum: From the e-mail account you intend to use, send a message to listserv@loc.gov containing the message: subscribe ead [Firstname Lastname].

that could deliver large and complex files in a variety of formats, including but not limited to SGML-encoded documents. (The Beinecke Library already had anticipated including preliminary and other nonstandard finding aids in ASCII text format along with its EAD instances.) For its part, the Yale EAD team wanted to provide the widest possible access to the SGML instances. This meant we would have to find an alternative to the proprietary software/server model embodied in the Dynatext/DynaWeb package that was being tested by other early implementers, including the Berkeley team.

Given our requirements, a package that included an OpenText server and SoftQuad's SGML authoring, publishing, and Web-browsing tools, including the freeware browser PanoramaFree, was the most logical choice. The OT server had the capacity to manage and deliver files in various formats, and we were told it was fully compatible with SoftQuad's products. A version of WordPerfect with SGML editing capabilities also was included in our start-up group of software: it would be tested by Divinity and by Manuscripts and Archives, who were already using WordPerfect to produce their finding aids, while Beinecke would experiment with SoftQuad's Author/Editor.

We have since discovered that vendor claims, however confident, do not always pan out quite as expected. ''Compatibility'' is a relative term, and local circumstances can influence just how well all the pieces fit together, in ways that cannot always be anticipated. And, yes, on occasion some features simply do not work as advertised, and you may be the first to point out a problem to the product developer! Similarly, the proliferation of SGML-enabled Web browsers, which many at the Berkeley conference predicted was just around the corner, has been slow to materialize. Even SoftQuad has wavered on its previous assurances that it would maintain a freeware version of its SGML Web browser.

But, as we also learned, all of this is part of the price one pays for being an early implementer. On the other hand, the members of the Yale task force acquired a much better understanding of the range of technical issues associated with a project of this type than would have been the case if things had gone as smoothly as we had originally envisioned. In the long run, although the learning curve was steep (and at times extremely frustrating) and numerous unanticipated hurdles had to be overcome, it was worth the effort and will stand us, and Yale, in good stead. In the meantime, we will continue to work with the hardware and software vendors, as well as with other interested parties in the archival and computer science fields, confident that in time the glitches will be resolved and products that fully meet our requirements will emerge. Fortunately, in this we are not alone.

Another of our key early decisions was that we would build the database as quickly as possible. We were determined that the Yale site would become an effective research tool and not languish as an innovative but not very useful oddity in the universe of online resources that we offer to researchers, students, and staff. As much as was possible, both legacy file conversion and the encoding of newly created files had to become priorities. We followed the advice of experienced SGML consultants, who encourage those facing massive legacy file conversions to keep the markup simple and to focus on tagging structure, not content.[4] This was not a difficult choice to make: we had considered both

[4]Briefly, structural markup is markup that codes the physical (and intellectual) structure of the document, such as series headings and cross references, the front matter (e.g., table of contents, biographical note, narrative description of the collection) and the container list (the box and folder inventory of contents). Generally, if a document conforms to a well-defined format or template, structural markup can be readily automated. Content markup, in contrast, focuses on the text of the document itself as the basis for locating specific occurrences of

the tagging and authority control issues associated with in-depth content tagging and were already inclined to take a "wait and see" approach to the question of content markup. We were, and remain, skeptical that the benefits to be gained by further enhancing access to specific data elements within the text of existing finding aids, beyond that already provided for in our local practice (e.g., using *AACR2* forms of headings for key names in lists of correspondents), will outweigh the effort involved and justify the allocation of scarce resources to such an indexing and authority control effort, particularly given the sizable backlogs of unprocessed and inadequately listed collections most of us face.

With these priorities in mind, we developed our markup tools and strategies to maximize automated functions and minimize the need for time-consuming manual intervention. We eventually rejected both SoftQuad's Author/Editor and WordPerfect's SGML editor, which we found cumbersome and particularly ill-suited to legacy file conversion; finding aid creation is an ongoing process of composition and editing, and we did not want to burden staff with the details of encoding at the same time they are writing a finding aid. All in all, we found it easier to stick with our current practices for creating finding aids and augment them by use of encoding macros or templates.

A third important early decision was to retain a measure of autonomy for each of the departments by providing for searches that take into account similarities and differences in finding aid practice, as well as in patterns of collecting and in research use of our collections. To this end, we developed both an all-Yale search form and unit-specific search forms, and we indexed the database in a parallel manner. Also, in the few cases where content markup varies by unit, the query boxes in the pull-down menus of the unit-specific search forms prompt those options. We also agreed that strict conformity to a single finding aid "look" or practice was neither desirable nor necessary, and as a result, each unit was allowed to design its own style sheets and navigators.

Local Finding Aid Practice and File Conversion Strategies

As previously stated, all three units were creating finding aids in machine-readable form when we embarked on the EAD project. Beinecke had approximately 250 finished finding aids, all in machine-readable form, that had been produced using a powerful DOS-based text editor, Edix/Wordix.[5] The Divinity Library and Manuscripts and Archives Department had been using WordPerfect for some time, but 50 percent of the Divinity Library's seventy legacy files and 90 percent of Manuscripts and Archives' estimated eighteen hundred legacy files still existed in paper form only. While the Divinity Library was able to convert its remaining thirty-odd paper files to WordPerfect in relatively short order using optical character recognition (OCR), the bulk of Manuscripts and Archives' files, including all of the paper-based legacy data, has yet to be converted.

a given proper name or corporate name, or to distinguish a date from other forms of numeric or alphanumeric combinations. Content markup is less readily accomplished using automated techniques such as macros; accuracy of markup, which is essential for meaningful indexing of content tags, usually requires human intervention to verify the meaning of the text and establish correct tag assignments. When content and structure converge (as in the Beinecke case, where folder dates always occur at tab 63 in a printed finding aid), tagging for content can be accomplished with macros with considerable accuracy and little or no need for manual review.

[5]This software package, which came out in the early 1980s, is no longer commercially available. It proved so effective for finding aid production, however, that Beinecke has continued to use it. We are investigating options for migrating to a more current software package; one that will allow us to preserve the inherent logic and functionality of our Edix/Wordix based system, while also employing a standard, nonproprietary programming language.

As we shall see, differences in the character and scope of the legacy files, combined with current practices for creating and maintaining finding aids in machine-readable form, had a decisive impact on the methods adopted for encoding both new and existing files, as well as on the rate at which legacy files could be converted. As much as anything, it is these differences that account for the fact that, from the outset, the majority of EAD instances on the Yale site have been Beinecke files.

Divinity Library and Manuscripts and Archives Department: From WordPerfect to Macros and Templates

During the summer of 1996 staff in the Divinity Library and Manuscripts and Archives Department successfully encoded a group of files using macros they had written for WordPerfect. They have since opted for a template-based method for both legacy file conversion and the encoding of newly created finding aids.[6] Each archives found that subtle (and not so subtle) differences in the structure of finding aids generated in their units made it impractical to take a direct macro-based approach to the encoding of files; a template proved to be a more efficient and reliable method for legacy file conversion and for encoding new files. Moreover, this work can be done by staff who have only a general awareness of SGML or the specifics of EAD, because the tagging is inserted "behind the scenes," as output from the template. Although the switch to a template has made conversion easier, the process remains fairly labor intensive and does not lend itself very well to the sort of batch treatment of legacy files that proved to be such an advantage for file conversion at Beinecke.

The Beinecke Library: Macro-based Conversion and Batch Processing of ASCII Files

The Beinecke has used sophisticated automated routines to create its finding aids since 1986. This has been possible because the structure of the finding aids conforms to a rigid format predicated on the placement of specified data elements at fixed tab locations. For example, box numbers flush left, folder numbers at 6, series names at 16, subseries at 18 and 20 (sometimes 22), folder descriptions at 22 (sometimes 24), notes at 24 (sometimes 26), and folder dates at 63 (see Figure 1).[7] Guidelines for formatting text within the data elements (such as how to handle wrap-around text in notes or to format date infor-

[6]For further information about the Divinity Library and Manuscripts and Archives Department EAD conversion templates, contact Martha Smalley, Research Librarian, Yale Divinity Library (martha.smalley@yale.edu) or consult the Yale Divinity Library's in-house training guidelines on the Web, <http://www.library.yale.edu/div/sgmanual/htm>.

[7]Summary of .box file tab specifications for Beinecke finding aids:

Flush left (tab 0).....for BOX NO.

Tab 6.....for FOLDER NO.

Tab(s) 16, 18, 20, (22)..for HEADINGS (series, subseries, sub subseries, and sub sub subseries).
Headings at 16, 18, and 20 should not extend beyond tab 70 and must not exceed one line in length. Headings at 22 should not extend beyond tab 60 but may extend beyond a single line.

Tab(s) 22, (24).....for FOLDER TITLES
Folder descriptions should not be placed at a tab value higher than 22; if tab 22 is functioning as a sub sub subseries heading, folder titles go at 24. Folder description text should not extend beyond tab 60.

Tab(s) 24, (26) .....for NOTES, CROSS REFERENCES
Notes and cross references should not be placed at a tab value higher than 24; if tab 24 is functioning as a folder title, notes and cross references go at 26. Do not set additional tabs beyond 26. Text of notes and cross references should not extend beyond tab 60.

Tab 63.....for DATE
Date information should not extend beyond tab 76.

Tab 78.....RIGHT MARGIN

REGISTER TEMPLATE

```
                    1 1 2 2 2 2                          6 6   7   7
1       6           6 8 0 2 4 6                          0 3   0   8
|       |           | | | | | |                          | |   |   |
                    Series heading_____
                        Note_____
                        Note_____
                        _____
                    SUBSERIES heading_____
                        Note_____
                      Sub subseries heading_____
                        Note_____
Box#    Folder#         Folder title_____                Dates____
        Folder#         Folder title_____
                        _____                            Dates____
                          Note_____
                          _____
        Folder#         Folder title_____                Dates____
                        _____
                          See also: cross-reference_____
                        Folder title_____                Dates____
                          See: cross-reference_____
                      Sub subseries heading_____
                        Sub sub subseries heading_____
        Folder#         24 Folder title_____             Dates____

        Folder#         24 Folder title_____             Dates____
                        _____
                          26 Note_____
                          26 See also:_____
```

Figure 1. Excerpt from the Beinecke Library Manuscripts Unit's *Microcomputer Manual for Registers*.

mation at the file level) further assure a very uniform structure, both within and across a wide range of finding aids. The layout of the front matter, though less structured, is equally precise.

In addition to prescribing the precise location and formatting of specified data elements, each of Beinecke's finding aids is made up of two ASCII text files: a front matter file ([filename].frn) and a box and folder list file ([filename].box); an optional appendix file ([filename].app) is provided for but rarely used. Staff process the two files through a series of locally defined Edix and Wordix programs and macros to manipulate the front matter file and box and folder list into a variety of useful forms: an output to screen of the document, stripped of diacritics, that is used in an in-house full-text database; a printer-

ready file, including diacritics; and sets of folder and box labels. A set of diagnostic macros tests for errors in data placement and in box and folder numbering; another macro automatically assigns folder numbers in sequence. Printed and on-screen reports of the diagnostics are output for staff review.

All of this data processing and various paper-based and machine-readable outputs are possible because the structure of the finding aid conforms to well-established and rigorously enforced specifications. When it came to doing SGML markup, it was therefore a relatively straightforward matter of building on the existing logic of our finding aid routines. We wrote new Edix/Wordix programs and macros to parse the .frn and .box files and to insert EAD tags at appropriate locations in each file structure. Some manual follow-up was necessary to review the markup, insert unique data (such as addresses for hypertext links within the document), or resolve problems, but this was a relatively minor task compared to the encoding of the files overall, many of which exceeded one hundred pages in length.

The basic division of front matter and container list into separate ASCII files had the added advantage of allowing us to batch the SGML markup and process large numbers of files in sequence with minimal manual intervention. Similarly, when we were preparing files for encoding, we were able to write macros to pinpoint formatting errors or other trouble spots (which early testing had identified as being problematic for automated markup) and to make the necessary corrections globally across dozens of files.

Because of the consistency of our file format and the capacity for batch processing, basic structural markup of Beinecke's 260 files, encompassing thousands of pages of text, was accomplished in a few days.[8] The container list markup (by far the largest and most complex portion of any finding aid) took only a few hours. The rest of the time was spent marking up the front matter; providing the necessary internal cross-reference links; reviewing, troubleshooting, and validating the files; and sending them to the server to be loaded and indexed.

Some Words of Advice for Others

With more than two years of work now behind us, what have we learned that might be of use to others who are considering embarking on an EAD project of their own? A great many things could be mentioned, but a few stand out as being particularly important.

If you have written guidelines and/or formatting templates for finding aids, review them, improve them if needed, and enforce them. If you don't have written guidelines, create them and enforce them. Similarly, once you determine your EAD-encoding practice and tag set, document it and keep it up to date![9]

Beyond formalizing and enforcing formatting guidelines for newly created machine-readable files, unquestionably the single most important piece of advice we can offer is to define your priorities. How much time, really, are you willing and able to devote to the project? Be realistic and set your goals accordingly; to get even a few files up and running

[8]A recent test run to encode a finding aid composed of a box and folder list with 245 file entries, plus standard front matter, took fifteen minutes. Basic encoding of the container list took ninety seconds; the remainder of the time was spent inserting internal hypertext links (e.g., cross references) using a combination of a special macro and manual markup, other follow-up work, verification, troubleshooting, validation, file naming, and exporting the file to the load queue on the server.

[9]To view the Beinecke Library's draft SGML/EAD Instructional Guidelines for staff, see: <http://www.library.yale.edu/beinecke/manuscript/sgmlmain.htm>.

may entail a significant investment in time, especially when you are just starting out and are still learning the ropes. Is legacy file conversion a priority? Or are you primarily interested in working EAD into your routines for creating new files? These are critical questions; your answers will determine how you proceed, the kind of resources you will have to commit, and the type of technical support you will require.

Remember, too, that the priority you are able to give to legacy file conversion may determine the utility of your on-line files for general research and reference use. If you have a significant body of legacy data to convert, you should therefore consider carefully the level of markup that you should attempt. The number of EAD tags that are *required* is about twenty (all of them structural, plus a few that are required to create a valid EAD instance). Twenty out of a tag library of 135 may not sound like much, and hardly worth the effort, but you might be surprised to learn that a very useful result can be obtained even with a minimal amount of tagging. If your finding aids have not previously been available on-line, even the simplest markup will be a great step forward. If you get through converting all of your data and feel the need to do more, there's nothing to stop you. You may well find, however, that a minimalist approach to EAD tagging is quite sufficient.

When planning legacy file conversion, carefully assess the degree of standardization present in those files. Wherever possible, find ways to group files of like type so as to take advantage of any potential that may exist for batch processing. Set your goals for legacy conversion, as compared to encoding of newly created files, by taking into account your levels of standardization; this will determine your capacity to convert files quickly. Don't set yourself up for defeat by aspiring to a rate of conversion that you cannot sustain given your resource base and/or the relative lack of standardization in your legacy data. As the Beinecke experience amply demonstrates, standards for data input are a key factor in being able to employ macros for markup; the more idiosyncratic the format of the finding aid, the more sophisticated your macros and programs must be, and the fewer your options for using macros at all.

If you anticipate having to develop your own markup macros or other tools rather than using existing SGML authoring tools, do you have the requisite technical knowledge for writing the macros and programs you will need? It seems highly unlikely that you will be able just to adopt a template (or macros) written for another institution's finding aids and, wholesale, apply them to your own; some tweaking, if not major rewriting, will almost certainly be required. If you do not have this expertise yourself, or elsewhere in house, you will have to obtain the assistance of someone with a fairly sophisticated understanding of the tools you propose to use for writing the macros. That person also must have, or acquire, a fairly sound understanding of your finding aid practice.

Finally, hardware is a consideration not to be underestimated, especially if your files are large and complex. The power of the CPU and the size of the display screen can make a world of difference in retrieving, navigating, and viewing files.[10] If, as we did, you plan to install a server, or if you will be working with new or unfamiliar hardware and software components, expect to spend more time than you anticipate dealing with the consequences of products that do not function as advertised (at least not in *your* systems environment) or that may not become available as projected. After all, EAD implementation, especially on the Web, in many ways is still a work in progress for us all.

[10]For our current recommended technical specifications, see "System Requirements for Panorama browsers" on the Yale Finding Aid Project website <http://webtext.library.yale.edu/#view>.

Conclusion

It is often said that ''Fools rush in where angels fear to tread.'' Certainly the Yale EAD early implementers have had reason to recall this saying on several occasions and to wonder in which camp we stood. As I have suggested, our EAD/Web server initiative proved to be a vastly more complex and time-consuming project than we had envisioned; the learning curve was steep and at times frustratingly slow. And as if our own inherent ''weaknesses'' of limited experience with SGML, Web publishing and the like were not enough to contend with, it seemed on more than one occasion that an unending stream of technical ''glitches'' and exceptions to the claims of functionality made by product vendors might defeat us. But in spite of all that, we prevailed, and there is general agreement that the experience has been worth the effort, and not just for the specific results embodied in the current Yale Finding Aid Project database and website. EAD, and Web distribution of SGML-encoded documents in general, are still undergoing development, and it is not yet clear just where we will arrive when all is said and done. What is fairly sure, we think, is that with this initiative the Yale library has entered into a new and powerful world of electronic and hypertext document delivery to expand awareness of and access to Yale's remarkable array of special collections, and that there is no turning back. We have achieved a hard won but critical first step in the right direction on behalf of ourselves, as library and archival professionals, for the collections entrusted to our care, and for the students, faculty, and research public who use these collections and who unquestionably will benefit from Web-based access to information about Yale's archival and manuscript holdings.

As to where we go from here, stay tuned....