
Price-Wilkin, John. "A Gateway Between the World-Wide Web and PAT: Exploiting SGML Through the Web." The Public-Access Computer Systems Review 5, no. 7 (1994): 5-27. To retrieve this file, use the following URL: gopher://info.lib.uh.edu:70/00/articles/e-journals/uhlibrary/pacsreview/v5/n7/pricewil.5n7. Or, send the following e-mail message to listserv@uhupvm1.uh.edu: GET PRICEWIL PRV5N7 F=MAIL.

1.0 Introduction

The HyperText Markup Language (HTML) used by the World-Wide Web has limited markup and structure recognition capabilities. Only a small set of text characteristics can be represented, and few of these have any functional value beyond display capabilities. The HTML ANCHOR element supports hypertext links; however, it cannot retrieve components of a linked document, such as a single glossary entry from a collection of several thousand entries, without resorting to programs external to HTML and the Web server. In spite of these limitations, HTML and the Web are key technologies for libraries.

The Standard Generalized Markup Language (SGML) is a full-featured, standard markup language. HTML is actually an SGML Document Type Definition. Ideally, it would be possible to retrieve text documents marked up with the richer SGML tag set via the World-Wide-Web.

This technical paper discusses how the Web can be linked to the PAT system, Open Text's search engine that supports access to SGML-encoded documents. This Web-to-PAT Gateway utilizes the Web's Common Gateway Interface (CGI) capability and SGML-to-HTML filter programs.

After briefly overviewing key technical concepts, the paper explains the operation of the Web-to-PAT Gateway, using several examples of how it is employed at the University of Virginia Libraries, including access to text files such as a Middle English collection, the Oxford English Dictionary, and the Text Encoding Initiative's Guidelines for Electronic Text Encoding and Interchange.

2.0 Key Concepts

This approach to using the Web to provide access to complex textual resources involves many tools and concepts that may be unfamiliar to the reader. This section provides a very brief overview of these complex topics and it describes their interrelationships.

2.1 SGML and HTML

Standards and open systems must be an essential part of library efforts to provide large-scale, wide-area access to textual resources. Textual resources must be reusable. Because of the cost of creating texts, it must be possible to use the texts in a variety of settings with a variety of tools. To that end, a standards-based encoding scheme must be the foundation of text

creation.
     The Standard Generalized Markup Language (SGML), an international standard, is such an encoding scheme, and it has proven extremely valuable in effecting an open systems approach with text. [1] This paper is not the place to present a detailed argument for using SGML, especially when this has been done so effectively elsewhere. [2] However, in addition to its value as an internationally approved standard, SGML is ideally suited to supporting text retrieval because it is a descriptive rather than a procedural markup language. SGML is a language designed to reflect the structure or function of text, rather than simply its layout or typography. In a text retrieval system, portions of an SGML document can be searched and retrieved, and functionally different textual elements can be displayed in accordance with their function.
     The difficulty of designing an implementation of SGML to meet a broad range of text processing needs in the humanities has been met by the Text Encoding Initiative (TEI) in its Guidelines for Electronic Text Encoding and Interchange. [3] The application of SGML using the TEI Guidelines will play a central role in ensuring that textual resources--particularly those important to textual studies--are produced in a way that make them flexible and of continuing value. The TEI itself is a collaborative project of the Association for Computers and the Humanities, the Association for Computational Linguistics, and the Association for Literary and Linguistic Computing. Its purpose is the promulgation of guidelines for the markup of electronic text for a variety of disciplines involved in the study of text. In mid-1994, a comprehensive and detailed two volume set of guidelines was published. The print version of the TEI Guidelines is an absolutely essential acquisition by libraries; an electronic version has been made available by the author. [4]

     A central feature of SGML is the DTD (Document Type Definition). The DTD is a codification of the possible textual characteristics in a given document or set of documents. SGML expresses the organization of a document without necessarily using the file system paradigm (i.e., discrete files representing the organizational components of a document). It expresses textual features (e.g., footnotes, tables, and headings) and the building blocks of content (e.g., paragraphs) using a descriptive language focusing on the role of the element, rather than some presumed display value. SGML is not a tag set, but a grammar, with the "vocabulary"--or tags--of an individual document being articulated in its DTD. Using this rigorous grammar, SGML can both declare information about the document in a way that can be transported with the document and can enforce rules in the application of markup by aiding in "parsing" the document.
     The HyperText Markup Language (HTML), which is used with the Web, is a form of SGML expressed by its own unique DTD. The shape of the HTML DTD has changed significantly since first articulated by researchers at CERN, and it continues to change with the demands of the Web. [5] HTML was designed to facilitate making documents available on the Web, and it expresses a variety of features such as textual characteristics and hypertext links. These hypertext links are HTML's most useful capability because they allow authors to link documents to other resources throughout the Internet, effectively making the Internet into a

large hypertext document.

2.2 CGI and FORM Use

The Web is far more than a server protocol for the transfer of HTML documents. Among the many resources it offers in facilitating sophisticated retrieval of information is the Common Gateway Interface (CGI). Like HTML, CGI is in transition. However, in its current state, it offers capabilities that allow the Web to support much more complex documents and retrievals than HTML alone supports. The Common Gateway Interface is a set of specifications for external gateway programs to speak to the Web's server protocol, HTTP. It allows the administrator to run external programs from the Web server in such a way that requests from the server return a desired document to the user or, more typically, generate a document on the fly. This capability makes it possible to provide uniform access to data structures or servers that are completely independent of the HTTP, including structures such as those represented in SGML documents or Z39.50 servers. The CGI specification is available on the NCSA documentation Web server. [6]

Closely associated with the CGI is the FORM specification, which was first introduced with NCSA's Mosaic Web client. This feature is a client-independent mechanism to submit complex queries, usually through a graphical user interface. FORM-compliant interfaces such as Mosaic, Lynx (a UNIX VT100 client), and OmniWeb (a NeXTStep client) use fill-out forms, check boxes, and lists to mediate queries between the user and CGI resources. Users respond by making selections that qualify submissions to the server (e.g., checking a box to indicate that a search is an author search) thereby making a complex command-line syntax unnecessary. [7]

2.3 Computer Languages and CGI

CGI programs can be written in a variety of languages, including UNIX shell scripts, C programs, and Perl. In fact, there are few limitations on the type of language that can be used. Perl is foremost among the options available to most Web administrators. Largely the work of Larry Wall, Perl can be used to create extremely fast and flexible programs with no practical limits on the size of the material it can treat. Perl also has outstanding support for the UNIX "regular expression," making it ideal for text systems where one form of markup must be translated to another. [8]

3.0 The Web-to-PAT Gateway

The modular approach taken in the Web-to-PAT Gateway separates the operations of retrieval to allow one component (e.g., an SGML-to-HTML filter) to be upgraded without affecting other components. It should be emphasized that this separation of operations grew out of local needs and that other approaches, including an approach that combines all operations in a single program, are possible. The four steps are:

1. FORM Handling

Users, with the aid of the FORM, submit a query.

2. CGI Query Handling

   The query is received and translated to a PAT search.

3. PAT Result Handling

   Information returned from PAT is transformed into lists or entries that can be selected.

+ Page 9 +

4. SGML-to-HTML Filtering

   The richer SGML is transformed into HTML.

This multi-stage approach has many advantages. For example, it is possible to use different programming languages or other software tools for each processing stage, selecting them based on their utility for particular functions or their ability to comply with local requirements. In the approach documented here, HTML FORMs, shell programs, C programs, and Perl programs have been used for the four operations. Separating the functions also allows persons with different responsibilities, skills, or interests to manage the different processes. For example, a system administrator might manage the second and third stages, while someone responsible for more aesthetic issues in the delivery might manage parts of the first and the fourth stages. At the University of Virginia Library, SGML-to-HTML filters continue to be enhanced by staff from the Library's Electronic Text Center in a process completely separate from the development of other parts of the interface.

3.1 HTML FORM for Query Submission

The use of an HTML FORM to handle query submission may be simple or complex. The three examples given here demonstrate that range: the Middle English FORM supports word and phrase searches; the Oxford English Dictionary search provides a great deal of information about the areas to be searched and information to be retrieved; and the TEI Guidelines FORM allows users to browse the document in a variety of ways, such as by chapter or other section. (The Middle English and TEI Guidelines resources are encoded in SGML.)

+ Page 10 +

3.1.1 Middle English Query

The FORM created for Middle English materials was deliberately made simple to allow users to retrieve keywords-in-context (KWIC) without knowing commands such as those needed to view search results. [9]

A search term is requested from the user and registered as the variable "query." So that neither the user or system is overwhelmed by large result sets, the size of result sets is limited to 100 items, and an additional FORM option (registering the variable "size") is included to help the user subsequently move through the results 100 items at a time or to sample 100 items from the entire result set.

3.1.2 OED Query

The richness of the Oxford English Dictionary (OED) is often overwhelming even for sophisticated users. Most users do not want keyword-in-context results and would prefer simple look up capabilities. The OED is a complex product designed to facilitate a broad array of activities. Consequently, even simple searches require elaborate query structures.

The OED FORM assists users in submitting many of the most commonly performed searches, including dictionary entry retrieval with simple look ups and truncated term look ups (e.g., "photo" for all words beginning with this stem). [10] It is also possible to retrieve quotations by the quote's author (e.g., retrieval of all quotations authored by Chaucer). This process includes the following:

1. In the FORM, the user submits a search term which is captured as the variable "query."

2. The user selects the type of search. Many types of searches are possible, including traditional look ups, alphabetic browses, full-text searches, and quotation retrieval.

3. Several other elements are used to limit the size of results. As in the Middle English search FORM, a default of no more than 100 results at a time may be viewed from each search.

4. In addition, a variable called "period" is offered to allow users to limit quotation searches by century.

3.1.3 TEI Guidelines Query

The structured browsing of the TEI Guidelines adds another important feature for mediating access to large or complex collections. Users of the TEI Guidelines are as likely to want to read a chapter or section as they are to want to search the contents.

To facilitate this sort of browsing, an initial HTML page is created containing the titles of the major SGML hierarchical structures of the TEI Guidelines (e.g., the DIV0 element), and each of these structures is linked to an HTML page containing the titles of subsidiary structures (e.g., the DIV1 through DIV4 elements). [11]

For example, the top-level HTML page is linked to the secondary HTML page for Part I of the TEI Guidelines. [12]

Figure 1 shows the top-level HTML page.

Figure 1. Top-level HTML Page for the TEI Guidelines

TEI Guidelines for Electronic Text Encoding and Interchange (P3)

You may also browse the Guidelines.

- Bibliographic header of the TEI Guidelines
- Preface

```
         * Acknowledgments
         * Changes from TEI P1 to TEI P3
         * Part 1: Introduction
         * Part 2: Core Tags and General Rules
         * Part 3: Base Tag Sets
         * Part 4: Additional Tag Sets
         * Part 5: Auxiliary Document Types
         * Part 6: Technical Topics
         * Part 7: Alphabetical Reference List of Tags and
                   Attributes
         * Part 8: Reference Material
```

Figure 2 presents the beginning of the HTML page for Part I of the TEI Guidelines.

Figure 2. Beginning of the HTML Page for Part I of the TEI Guidelines

----

```
Part I: Introduction

     * 1: About these Guidelines
          + 1.1: Structure and Notational Conventions of this
                 Document
               o 1.1.1: Structure
               o 1.1.2: Notational Conventions
          + 1.2: Underlying Principles and Intended Use
               o 1.2.1: Design Principles of the TEI Scheme
               o 1.2.2: Intended Use
                    # 1.2.2.1: Use in Text Capture and Text
                               Creation
                    # 1.2.2.2: Use for Interchange
                    # 1.2.2.3: Use for Local Processing
          + 1.3: Historical Background
               o 1.3.1: Origin and Development of the TEI
               o 1.3.2: Future Developments

     * 2: A Gentle Introduction to SGML
          + 2.1: What's Special about SGML?
               o 2.1.1: Descriptive Markup
```

The URL for each list item in the Part I page contains the information necessary to conduct a search and retrieve the structural component being selected. For example, to retrieve the section "Structure and Notational Conventions of this Document," the first subsection of the first chapter in Part I, the URL points to the component extraction program tei-tocs, and it specifies that this is an ID "struct" at level DIV2 (e.g,. the section is bounded by the tags <DIV2 ID="struct"> and </DIV2>).

3.2 CGI Query Negotiation

The CGI query handling operation accepts the search or action initiated by the user and prepares it for submission to PAT. [13] Information from the user is submitted to the CGI program as a FORM or URL (e.g., <FORM ACTION=/bin/tei.sh>). A wide range of queries can be supported using this strategy. For example, a single word or phrase search can be submitted; it is also possible to retrieve two words within a specified proximity to each other, two structures (e.g., stanzas) that contain one or more words or phrases, or structural units such as chapters. UNIX Bourne shell scripts are used to interact with PAT. The program begins by testing to see if a query has been made, and preliminary PAT results are tested to determine whether a valid search was performed.

3.2.1 Middle English Query

In the Middle English query, simple searches are accepted and KWIC results (64 characters in each) as well as a citation are sent to the user's screen. The "citation" is actually an ID assigned to each text and extrapolated throughout the text, page by page, by Electronic Text Center staff. For example, the ID "JefLett" is assigned to Jefferson's Letters, and, on page 234 of the document, the ID will be "JefLett234." This ID value becomes a hypertext link for viewing the larger document context in the subsequent processing of results.

The PAT segment of the program for the Middle English collection [14] consists of commands that:

1. Move PAT into its "QuietMode" for tagged communication with another program ({QuietMode raw}). (See section 3.3 for further discussion of QuietMode.

2. Set the sort order and direct PAT to use the ID described above as the identifier for each result ({PrintMode 3 ID page}).

3. Search the word or phrase.

4. Print the set of results requested by the user (e.g., first hundred items).

3.2.2 OED Query

In the OED query, a more sophisticated query that takes advantage of PAT's structure recognition capabilities is submitted. The CGI script for the OED uses several structures based on tags in the OED to search this text file and generate results using PAT. The searches submitted by the OED FORM are created using shell program case statements. [15]

For example, if the user submitted a search asking for quotations authored by Bailey in the sixteenth century, the following query is submitted to PAT: docs C16 incl ("bailey" within docs A). This query retrieves document structures that are 16th century quotes that contain "bailey" within the quote author field.

A search ends with the appropriate "print" command to PAT, such as a command to print quote document structures (called Q structures).

3.2.3 TEI Guidelines Query

The approach taken with the TEI Guidelines query represents the richest implementation of this approach. It combines searching and browsing, and it enables browsing through the recognition of document structure. The CGI communication with PAT is also surprisingly simple. URLs are either static, embedded in the menu options, or they are generated dynamically as the result of a search (e.g., the TEI filter). [16]

In the static URL strategy, the two elements needed to conduct the PAT search (DIV level and ID value) are embedded in each URL. For example, the following URL sends the DIV level (<DIV3>) and the ID value (ABTEI2) to the CGI script tei.sh: http://etext.virginia.edu/bin/tei-tocs?div=DIV3&id=ABTEI2. [17] Based on this information, the tei.sh script formulates the following PAT query: docs DIV3 incl "<DIV3 ID=ABTEI2>." The script then issues the "print" command (pr.docs.DIV3) to retrieve the relevant portion of the TEI Guidelines.

3.3 PAT Result Handling and QuietMode

Two stages of result handling, each relying on PAT's QuietMode, take place before pages or dictionary entries are presented to the user. In QuietMode, PAT produces very helpful results with all information tagged for more reliable processing. For example, the size of the result set is marked with <SSIZE> tags (e.g., <SSIZE>23</SSIZE>), and every result can be made to begin with a <HDR> tag (e.g., <HDR>AusEmma234</HDR>). [18]

The first stage produces only minor transformations, primarily displaying the number of results retrieved and separating each result into a separate line. The results of the first stage are piped to the second stage to produce a user display.

The first-stage filters for the three example documents perform the following actions:

- Middle English: the filter uses <SSIZE> tags to highlight the number of results. [19]

- OED: the filter ensures that each result is on a separate line, deferring issues such as using the <SSIZE> tags to highlight the number of results until a later stage. [20]

- TEI: the filter is essentially the same as that used for the Middle English collection. [21]

The second stage presents each element to the user in a sort of index view so that a broader display can be produced. The Perl code for this stage usually produces a KWIC view, with each line being a hypertext link to an expanded view of the result. The user sees the HDR element content as the link, but the byte offset--the number of characters into the file that PAT uses to locate results--is the actual search component of the link.

The Perl code for the Middle English query is an example of the code needed to produce a KWIC view. It converts each result to a view and associated hypertext links. [22]

Results from the OED are presented to the user as a list of dictionary headwords, from which the entire entry can be displayed. [23]

3.4 Filtering SGML to HTML

The results of searches, each in a complex SGML form designed to support the Web-to-PAT Gateway, are prepared by the final processing stage for presentation to a Web client (e.g., Mosaic) after being filtered to HTML. Filtering, in these examples, is again achieved by Perl. Most SGML tags from the originating files will be specialized and they will not have a corresponding HTML tag. For example, the quoted author element in the OED (the <A> tag) has no corresponding HTML tag, but one might render it as italicized text. Because of this lack of correspondence and the limited number of HTML tags, decisions are largely arbitrary and draw on presentation or aesthetic needs.

The Perl program oed.pl is used to filter OED tags to HTML. [24] This filter and the resulting output demonstrates the challenge of filtering rich, heterogeneous text to HTML. A sample tagged OED entry is included in Appendix C to illustrate the problems encountered. Despite many compromises in mapping complex tags to simple presentation characteristics, information can be presented attractively to a user with a GUI client. Future improvements in HTML and the ability of clients to interpret a variety of tags should enhance the display of OED entries.

Subsequent to the KWIC view for TEI and Middle English queries, more thorough transformations are performed on the texts. For example, upon being selected, a result's byte offset is sent to a program that uses PAT to print a 1,500 character context for a result. This character context may contain any of the possible tags in the DTD, so a filter that represents all possible element values is created. At this time, the filters fall short in many areas, in particular in their ability to express complex relationships made possible in the DTD. For example, an element may have an attribute value suggesting that it should be tagged one way (e.g., "gloss" for glossary), as well as having sub-elements of the same type with different attribute values (e.g., "bullets"). For example, one might encounter the partial character context shown in Figure 3.

Figure 3. Example of Complex Element Attributes

```
<LIST TYPE="gloss">
          <LABEL>name of item 1
          <ITEM>contents of item 1
          <LABEL>name of item 2
          <ITEM>
          <LIST TYPE="bullets">
               <ITEM>contents 1 of item 2
               <ITEM>contents 2 of item 2
               <ITEM>contents 3 of item 2
          </LIST>
</LIST>
```

At the present time, the simple stream-oriented filters cannot differentiate between tags with the same name (e.g., the <LIST> tags) in this type of complex nesting relationship. Two example filters have been made available by the author. [25]

+ Page 17 +

4.0 Preparing Texts with PAT

Little or no preparation of SGML texts is necessary to be able to implement the Web-to-PAT Gateway. In order to provide a useful indicator of location for KWIC views, we typically add an ID element with concise positional information. For example, in our Modern English Texts, we add a combination of author, title, and pagination to create an ID such as JefLett244, which represents Thomas Jefferson, Letters, page 244. These ID values are limited to ten characters.

Since the TEI Guidelines use "minimization" (i.e., omit many end tags where they are implied by context), preparation of the TEI Guidelines was more complex, and it included the following steps:

- Normalization

  A parser was used to normalize the text. This resulted in all elements being closed and being put in uppercase. For example, if a chapter begins with the <div1 id=AB> tag and has no end tag, the initial tag is changed to <DIV1 ID="AB"> and the chapter is closed with a </DIV1> end tag.

- Added <ID> Values

  Three DIV1 elements did not have ID attributes. These were added to aid in retrieval. Attributes such as ID="bibliog" for the bibliography were also included.

- ID Element

  An ID was created for each DIV1, with the value of the ID attribute being copied to the ID. For example, for the chapter called "A Gentle Introduction to SGML," the <DIV1 ID="SG"> now has <ID>SG</ID>. This makes it possible to display the ID value for each DIV1 in the KWIC views.

+ Page 18 +

5.0 Testing

The Web-to-PAT Gateway has been employed to access text collections at a number of institutions, including the University of Chicago, the University of Michigan, and the University of Virginia. Open Text has generously allowed the University of Michigan and the University of Virginia to provide other sites with access to selected PAT-based collections or resources. To examine example components of the Web-to-PAT Gateway, use the test page at the following URL: http://www.lib.umich.edu/hti/.

6.0 Conclusion

The Web-to-PAT Gateway is an effective method for accessing text collections, and it suggests important possibilities for accessing other types of resources. [26]

The University of Virginia provides access to many collections of resources using clients designed to support complex analysis, where users can create sets, combine them, and use a range of operations facilitated through a wide array of menus. These clients are frequently much more complicated than is desirable for simple operations, such as word look ups in the OED. The Web-to-PAT Gateway allows users to do simple word look ups in the OED or to formulate simple queries in the text collections without needing to understand PAT syntax or the organization of the collections.

The Web-to-PAT Gateway strategy could also be applied to other types of texts, such as SGML-encoded journals. For example, a journal run marked up according to the more elaborate AAP (Association of American Publishers) DTD (ISO 12083) could return articles to the user using simple PAT queries. Another approach might set up CGI scripts to facilitate browsing where a user selects a browse by author/title option and is given a series of choices that use PAT queries to produce a menu of either journal titles, volume/issue numbers, or author/title options. The Web can be an effective means of accessing the original files in a fuller SGML markup without altering the markup or resorting to fragmenting the material into thousands of files corresponding to the individual articles or even parts of articles. Similar strategies for books and documentation can be imagined as well.

The Web-to-PAT Gateway offers significant evidence that SGML can serve as a delivery format for electronic materials sold by publishers to libraries and that SGML can serve as an effective internal file format on a library server.

Notes

1. Information Processing--Text and Office Systems--Standard Generalized Markup Language (SGML) (Geneva: The International Organization for Standardization, 1986).

2. Perhaps the best argument to date for the use of SGML in literary and linguistic computing is the often reprinted article: James H. Coombs, Allen H. Renear, and Stephen J. DeRose, "Markup Systems and the Future of Scholarly Text Processing," in The Digital Word: Text-based Computing in the Humanities, eds. George P. Landow and Paul Delany (Cambridge, MA: MIT Press, 1993), 85-118. Also very valuable is the thorough guide to the standard itself: Charles F. Goldfarb, The SGML Handbook (New York: Oxford University Press, 1990).

3. The first preliminary draft of the TEI Guidelines was published in 1990. The second draft (known as P2) was subsequently published in fascicles. It is available at the following URL: ftp://ftp.ex.ac.uk/pub/SGML/tei/p2. A complete, revised edition (P3) was published in 1994: C. M. Sperberg-McQueen and Lou Burnard, eds., Guidelines for Electronic Text Encoding and Interchange (Chicago: Text Encoding Initiative, 1994).

4. URL: http://words.hh.lib.umich.edu/TEI/.

5. The specifications for HTML continue to be in draft form, but they are being discussed by the Internet Engineering Task Force. A current version of the draft specification can be found at the following URL: file://info.cern.ch/pub/docs/html-spec.ps.Z.

6. Current CGI specifications and very useful examples can be found at the following URL: http://hoohoo.ncsa.uiuc.edu/cgi/overview.html.

7. FORM options and implementation examples are documented by NCSA at the following URL: http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html.

8. Larry Wall and Randal L. Schwartz, Programming Perl (Sebastopol, CA: O'Reilly & Associates, 1991).

9. The form can be seen at the following URL: http://words.hh.lib.umich.edu/ME.html. Other searches are facilitated as well, including proximity searches and traditional Boolean searches. In contrast to the simple search discussed in this article, these searches mediate user communication by taking multiple inputs (e.g., "love" and "envy"), an operator (e.g., near or intersect), and, in the case of the Boolean search, a document structure to use for intersection (e.g., a poem or chapter). This allows support for queries such as love near envy or stanzas that include love intersected with stanzas that include envy.

10. URL: http://words.hh.lib.umich.edu/oed.html.

11. The menus were created using a combination of manual and automated processes; the latter using PAT to identify components and programs (e.g., Perl programs) to convert output to HTML and URLs. This approach was selected in order to provide a static view of the organization of the TEI Guidelines. It should be stressed, however, that a completely automated approach is possible and may be incorporated in subsequent projects. In a more automated approach, something like the following would be performed:

1. User selects a TEI page, which initiates a PAT search returning all DIV0 elements HEAD. (For example, one might request docs HEAD within docs DIV0 if the HEAD element were unique.)

2. The information from the first step enables the dynamic construction of a menu option containing a URL that would extract HEAD items at levels DIV1 through DIV4.

3. The information about lower levels within each part could then be used to create URLs such as those described earlier (e.g., http://etext.virginia.edu/tei-tocs/div=DIV1&id=ABII).

12. URLs: http://words.hh.lib.umich.edu/TEI/ and http://words.hh.lib.umich.edu/TEI/tei-tocs1.html.

13. It should be emphasized that in this and many other stages, extensive interaction with the file system takes place, and every precaution should be taken to ensure security and protect against unpredictable results. Even though CGI/httpd implementors have taken precautions against dangerous queries, the httpd server should be run as "nobody" (or a comparable user) to protect against malicious queries. Also, note the precautions taken against communication errors with PAT and the prevention of runaway processes.

14. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/me-kwic.

15. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/oed-kwic.

16. See tei.pl available at the following URL: ftp:// sansfoy.hh.lib.umich.edu/htdocs/www-to-pat/tei.pl.

17. URL: ftp://ftp.lib.virginia.edu/www/staffpubs/tei.sh.

18. In most cases, a step is introduced to remove all newlines, conducted by piping output to the "tr" command (e.g., | tr -d "\012"). This is done to aid Perl in treating tag content that crosses boundaries. For example, the OED filter creates output where the results are rendered as a series of single lines, one for each result. It is not possible to remove the newline characters in the TEI Guidelines because many of the examples rely on formatting with newlines in CDATA elements. This results in inadequacies in the filtering where, for example, the beginning and tags are separated by several lines and the corresponding HTML are best determined by an attribute value (e.g., TYPE="gloss"). However, this is a shortcoming in the author's Perl skills rather than a fault in the TEI markup.

19. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/me-parse.pl.

20. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/oed-parse.pl.

21. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/teiquiet.pl.

22. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/me-kwic.pl.

23. This Perl code is available at the following URL: ftp:// sansfoy.hh.lib.umich.edu/htdocs/www-to-pat/oedkwic-filt.

24. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/oed.pl.

25. URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/ota.pl. See also a filter for the TEI P3 DTD at the following URL: ftp://sansfoy.hh.lib.umich.edu/htdocs/ www-to-pat/tei.pl.

26. There is still a compelling need for more specialized tools to access the same collections. Persons involved in complex analytical activities need to be able to combine sets in a variety of ways, to see numbers associated with previous searches at a glance, and to create and access structures not otherwise accessible through the Web-to-PAT Gateway. Currently the University of Virginia supports a locally developed VT100 client as well as commercial X Window System and Microsoft Windows clients from Open Text.

Appendix A. Proposed Enhancements

The following enhancements are proposed for the Web-to-PAT Gateway:

1. Incorporate PAT API

   Incorporating the Open Text Application Programming Interface (API) tools will make it possible to access PAT directly from the CGI programs, rather than creating a separate search file. This will be more efficient, will make possible testing for search size without re-executing the search, and may make possible a degree of interaction with the text not currently available.

2. Use Perl and Subroutines

   Perl is probably the most efficient programming language to mediate interaction between a FORM and PAT, and it can execute the separate components of the Web-to-PAT Gateway outlined in this paper as subroutines.

3. Improve SGML Filters

   If filters can be written to recognize the full range of relationships made possible by the SGML, filtering could be more attractive. For example, nested LIST could be differentiated both by the nesting and by the attribute values. Until a filter can be written to exploit these characteristics, it will fall short of what is possible.

Appendix B. OED Implementation Cookbook

This Appendix assumes the reader is familiar with the installation and configuration of the Web server (i.e., httpd). It is also extremely useful to have a familiarity with PAT and the OED.

1. Install the Web server (i.e., httpd). This implementation has only been tested with the NCSA httpd, available at the following URL: ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/current/httpd_source.tar.Z.

2. The locations of CGI scripts and the PAT executables are registered in the httpd's srm.conf file. For example, if we were put the CGI scripts and executables in the directory /patbin, the entry might read:

```
ScriptAlias /patbin/
/usr/local/httpd/patbin/
```

3. Access to the OED must be restricted to persons at your institution. For the NCSA httpd, this can be done by creating a file called .htaccess in the /patbin directory and declaring an acceptable set of IP addresses. See the httpd documentation for further information. Note that it is not important to restrict access to the HTML FORM oed.html, but it is to restrict access to the executable files.

4. A data dictionary, probably called 2e.dd, will have been included with your copy of the OED. Copy this data dictionary (e.g., copy 2e.dd to 2eWWW.dd) and in the new dictionary change the names of the "Regions" so that the simpler tag names are used. For example, change the name Entry to E.

5. Put the following files in the /patbin directory:

   o oed-kwic

   o oedkwic-filt

   o oed-parse.pl

   o kwic.input.sed

   o oed.pl

   o oed2html.pl

   o oed-id

   o oed.sh

6. Make certain that all the files in the /oedbin directory are executable.

7. Change the relevant portions (paths and variables such as titles) in the first lines of oed-kwic, oed-id, and oed.sh.

8. Alter oed.html to point to the correct path (e.g., /patbin), and alter oed.pl and oedkwic-filt to change /oedbin to the correct path (e.g., /patbin).

9. Check the location of Perl in your system and change all of the Perl scripts to conform to this location.

Appendix C. Sample Tagged OED Entry

```
<E><HG><HL><LF>debug</LF><SF>debug</SF><MF>debug</MF></HL><MPR>
d<i>i&mac.</i>b<i>&reva.</i>&sd.g</MPR><IPR><IPH>di&lm.&sm.b&
revv.g</IPH></IPR>, <PS>v.</PS></HG><ET>f. <XR><XL>de-</XL>
<SN>II</SN>. <SN>2</SN></XR> +<XR><XL>bug</XL><PS>sb.</PS>
<HO>2</HO></XR></ET><p><S4><#>1</#><S6><DEF><PS>trans.</PS>  =
<XR><XL>delouse</XL><PS>v.</PS></XR></p></DEF><QP><Q><D>1960</D>
<A>J. Stroud</A><W>Shorn Lamb</W> vi. 70 <T>We'll..take them
round to the Clinic, and..get them debugged there.</T></Q></Q>
</S6></S4><p><S4><#>2</#><S6><DEF><LB>slang.</LB>To remove
faults from (a machine, system,etc.).</p></DEF><QP><EQ><Q>
<D>1945</D> <W>Jrnl. R. Aeronaut. Soc.</W> XLIX.  183/2

<T>It ranged from the pre-design development of essential
components, through the stage of type test and flight test
and `debugging' right through to later development of the
engine.</T></Q></EQ><Q><D>1959</D><W>New Scientist</W> 26

Mar. 674/1 <T>The `debugging' time spent in perfecting a
non-automatic programme.</T></Q><Q><D>1964</D> <W>Discovery</W>
Oct. 51/3 <T>This failure report plays a vital role in the
process by which the scientist corrects or de-bugs his
programme.</T></Q><Q><D>1964</D> <A>T.  W. McRae</A>
<W>Impact of Computers on Accounting</W> iv. 99 <T>Once we have
`debugged' our information system.  </T></Q><Q><D>1970</D>
<A>A. Cameron</A> et al. <W>Computers &amp. O.E.

Concordances</W> 49 <T>Program translation, debugging, and
trial runs of the concordance were performed at the University
of Michigan Computer Center.</T></Q><Q><D>1970</D> <A>A.

+ Page 26 +

Cameron</A> et al. <W>Computers &amp. O.E.  Concordances</W>,
49 <T>By Christmas the program was debugged.</T></Q></QP></S6>
</S4><p><S4><#>3</#> <S6><DEF>To remove a concealed microphone
or microphones from (a room, etc.); to free of such listening
devices by electronically rendering them inoperative.  Cf.
<XR><XL>bug</XL><PS>sb.</PS><HO>2</HO><SN>3</SN><SN>f</SN></XR>.
orig.<LB>U.S.</LB></p></DEF><QP><Q><D>1964</D> <W>Business
Week</W> 31 Oct. 154 (<W>heading</W>) <T>When walls have ears,
call a debugging man.</T></Q><Q><D>1964</D><W>Business Week</W>
31 Oct. 154 (<W>heading</W> 158/2 )<T>He quotes high fees for
his work, saying that debugging equipment is expensive.</T></Q>
<Q><D>1966</D> in Random House Dict. </Q><Q><D>1969</D><W>New
Scientist</W> 16 Jan. 128/3<T>`Debugging' the boardroom and the
boss's telephone may become as common in industry as in the
unreal world of the super-spy. </T></Q><Q><D>1976</D><A>M.
Machlin</A> <W>Pipeline</W> xxxi. 353 <T>The room..had steel
walls and had been rigorously de-bugged.</T></Q><Q><D>1978</D>
<W>Sunday Mail Mag.</W> (Brisbane) 9 Apr. 3/6 <T>Jamil,
America's leading `debugging' expert, discovered the secret
of an exported `bug' which should not have worked.</T></Q>
<Q><D>1987</D><W>Daily Tel.</W> 3 Apr. 1/8 <T>American
officials are scrambling to `de-bug' their embassy in Moscow
before the arrival of Mr Shultz, Secretary of State, on Monday
week.</T></Q></QP></S6></S4><p><S4><SE>Also <BL>
<LF>debugging</LF><SF>de&sm.bugging</SF><MF>debugging</MF></BL>
<DEF><PS>vbl. sb.</PS> (see senses 2, 3 above).</DEF>
</SE></p></S4></E>
```

Acknowledgements

The author would like to express his deep gratitude to others who have assisted in the process of making the Web-to-PAT Gateway possible. In particular, I am especially indebted to Mark Olsen of the ARTFL project, who contributed his OED filter and provided guidance in establishing communication between the Web and PAT, and to Hannah Kaufmann of Princeton University and Peggy Yetter of the University of Pennsylvania for their persistence in working through early versions of the Gateway.

Thanks also to Charles Bailey for his unflagging editorial assistance with this paper.

About the Author

John Price-Wilkin, Coordinator of the Humanities Text Initiative of the University of Michigan Library, the School of Library and Information Studies, and the University of Michigan Press, 309 North Hatcher Graduate Library, University of Michigan, Ann Arbor, MI 48109-1205. Phone: (313) 764-8074. Internet: jpwilkin@umich.edu.

The Public-Access Computer Systems Review is an electronic journal that is distributed on the Internet and on other computer networks. There is no subscription fee.

To subscribe, send an e-mail message to listserv@uhupvm1.uh.edu that says: SUBSCRIBE PACS-P First Name Last Name.

This article is Copyright (C) 1994 by John Price-Wilkin. All Rights Reserved.

The Public-Access Computer Systems Review is Copyright (C) 1994 by the University Libraries, University of Houston. All Rights Reserved.

Copying is permitted for noncommercial use by academic computer centers, computer conferences, individual scholars, and libraries. Libraries are authorized to add the journal to their collection, in electronic or printed form, at no charge. This message must appear on all copied material. All commercial use requires permission.
