

# THE ARCHWAY PROJECT: ARCHITECTURE FOR RESEARCH IN COMPUTING FOR HUMANITIES THROUGH RESEARCH, TEACHING, AND LEARNING

KEVIN KIERNAN<sup>1</sup>, JERZY W. JAROMCZYK<sup>2</sup>, ALEX DEKHTYAR<sup>3</sup>, AND DOROTHY CARR PORTER<sup>4</sup>, WITH KENNETH HAWLEY<sup>5</sup>, SANDEEP BODAPATI<sup>6</sup>, AND IONUT EMIL IACOB<sup>7</sup>

<sup>1</sup>*University of Kentucky: English (kiernan@uky.edu)*; <sup>2</sup>*Computer Science (jurek@cs.uky.edu)*; <sup>3</sup>*Computer Science (dekhtyar@cs.uky.edu)*; <sup>4</sup>*Research in Computing for Humanities (dporter@rch.uky.edu)*; <sup>5</sup>*English (kchawl0@uky.edu)*; <sup>6</sup>*Computer Science (sboda2@.uky.edu)*; <sup>7</sup>*Computer Science (ionut@ms.uky.edu)*

Send correspondence to:

Dorothy Carr Porter, Program Coordinator  
Collaboratory for Research in Computing for Humanities  
3-51 and 3-52 William T. Young Library  
University of Kentucky  
Lexington, KY 40506-0456  
859-257-9549  
*dporter@rch.uky.edu*

**Abstract.** An unusual alliance called the ARCHway Project is developing an Edition Production Technology (EPT), a technological infrastructure for collaborative research, teaching, and learning between computer scientists and specialists in Old English.<sup>1</sup> Our goal is to identify and solve problems of mutual importance in building image-based electronic editions of significant cultural materials. The EPT will allow us to implement and integrate both new and already available software applications, to construct a digital library of

previously unedited Old English manuscripts as a testbed for our solutions, and to distribute the digital library to the public. This paper introduces some of the tools and technologies currently under development as we build this interdisciplinary research, teaching, and learning infrastructure. First, we introduce the stand-alone tools developed under the *Electronic Boethius* project, an image-based electronic edition of the Alfred the Great's Old English translation of Boethius's *Consolation of Philosophy*.<sup>2</sup> Next, we describe formal methodologies for collaborative research, teaching, and learning and the integration of these tools, as well as new developments, into an open-source platform. Next we present new ways to maintain the integrity of highly complex, layered, XML markup.<sup>3</sup> Finally, we discuss how the EPT will be useful to other humanities computing projects.

**Keywords.** architecture; concurrent hierarchies; database management; Eclipse platform; editorial tools; image-based editions; integrated tools; research, teaching, and learning; XML markup

## 1. The *Electronic Boethius* Project

Alfred the Great's Old English translation of Boethius's *Consolation of Philosophy* survives in two manuscripts, one from the tenth and one from the twelfth century, and in an indispensable seventeenth-century transcript and collation of the earliest manuscript, which was badly damaged by fire in the eighteenth century.<sup>4</sup> The confusing manuscript tradition, the catastrophic damage to the principal manuscript, and the easy recourse to the late transcript, have conspired to lead modern editors into producing two major editions, one in prose and one in verse, neither one remotely approximating the manuscript testimony.<sup>5</sup> An image-based electronic edition makes it possible to restore much of the damaged tenth-century manuscript and produce the first prose-and-verse edition of it.

British Library MS Cotton Otho A. vi, the earliest manuscript of Alfred's *Boethius*, is the principal manuscript tested for the ARCHway Project.<sup>6</sup> Using digital images from the *Electronic Boethius*, we will develop the EPT to encode searchable glossaries, transcripts, and editions, all linked to the images. Users at all levels will thus have unprecedented access to high-resolution representations of the damaged and fragmentary folios that survive. The ultimate goal, however, is to produce an EPT to serve as a generic workbench for any image-based electronic edition, regardless of language, date, or provenance.

### 1.1 THE ORIGINAL TOOLS

An early stand-alone version of the Glossary Tool began under the Digital Atheneum project<sup>7</sup> to allow the humanities editor and assistants to sort all words with locations (manuscript folio and line and edition line numbers), then to parse and define all parts of speech. Only then was it possible to make sense of the damaged manuscript folios and thus to provide useful data for computer scientists designing a database. Under the *Electronic Boethius* we redesigned and reprogrammed this prototype Glossary Tool to provide comprehensive XML tagging while working in the context of the text being glossed; we also enhanced the functionality of the tool by linking the text to its corresponding wordlist tree.<sup>8</sup> See Figure 1 for a screen shot of the Glossary Tool.

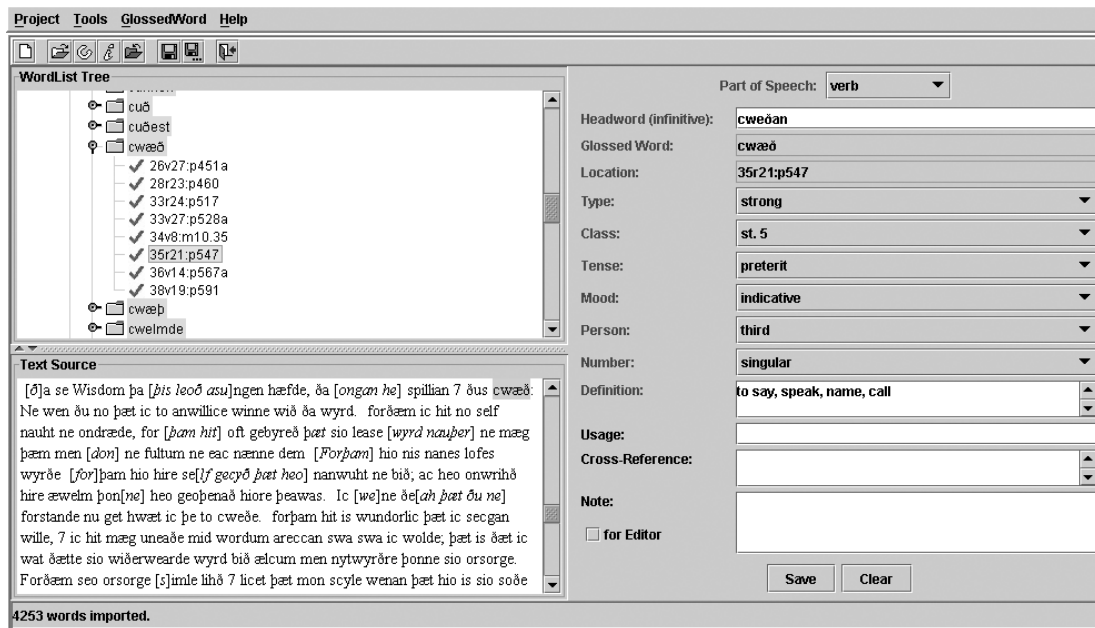


Figure 1: Glossary tool.

Using custom-made templates for each part of speech, we compile a glossary, including every instance of a word within a given text, in this case Book Two of Alfred's *Boethius*. These templates permit us to perform complex, exhaustive, XML tagging without ever having to see a tag. The resulting XML-encoded glossary is thus fully searchable, and may be reformatted for HTML or any other kind of useful display through XSL transformations.

It was clear from the start that existing software did not support the complex markup and text/image linking that was the heart of the *Electronic Boethius*. We therefore proceeded to develop a transparent method for XML tagging of a manuscript transcript in the context of images of the damaged manuscripts (see Figure 2). This complementary Tagger tool allows the editor and research assistants to provide pervasive, extremely complex, XML encoding for the transcript and edition, again without the necessity to consult the encoded XML file (there is also a window, closed

by default, which allows the curious user to examine the XML file with all the angle brackets).

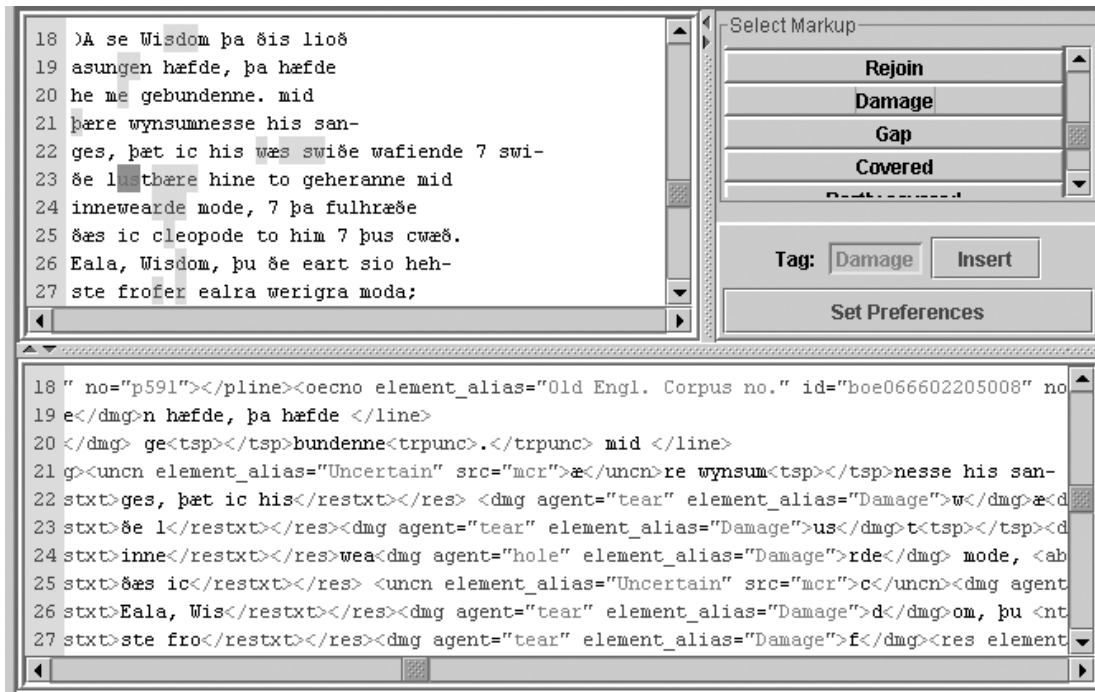


Figure 2: Tagger tool.

The Tagger and Glossary tools were incorporated, along with an XML validator and XML transformer, into a program called the Electronic Editions Editor (E3).

The custom-made software, specifically designed to relieve scholars from having to concern themselves with complex XML markup and its ever-growing forest of angle brackets, correctly nests all tags behind the scenes, silently avoiding the creation of invalid or non-well-formed XML encoding. In one window, the editor views the ultimate source of all markup – the manuscript folio – and in another window the editor tags a transcript using clickable element buttons, based on the given DTD. The tagging covers everything from paleographical features to minute physical

description of the fire damage; from technological restorations with special lighting to variant readings from other manuscripts or editions; from editorial conjectural restorations to editorial emendations. The resulting tagged file is, like the glossary, fully searchable and open to any number of displays. By including coordinates for all tagged parts of an image, XML and image searching proceed simultaneously.

Other programmers were hired under the *Electronic Boethius* project to reprogram the search facilities of the *Electronic Beowulf* and to develop editorial tools that integrated the high-resolution images with the XML tools or otherwise brought the images into the foreground.<sup>9</sup> The ScripText tool refined the Tagger by integrating the markup of image and text, allowing simultaneous searching of them.<sup>10</sup> A subset of the ScripText tool designed for the study of paleography provides templates that help the editor easily prepare searchable paleographical descriptions of the scribal letterforms by using easily amendable, extensible templates for all letters. X/Y coordinates are automatically included in the XML markup. These individually tagged letterforms can provide students with easy access to paleographical illustrations, which are usually omitted in scholarly editions and sparsely included even in paleographical treatises [fig. 3].

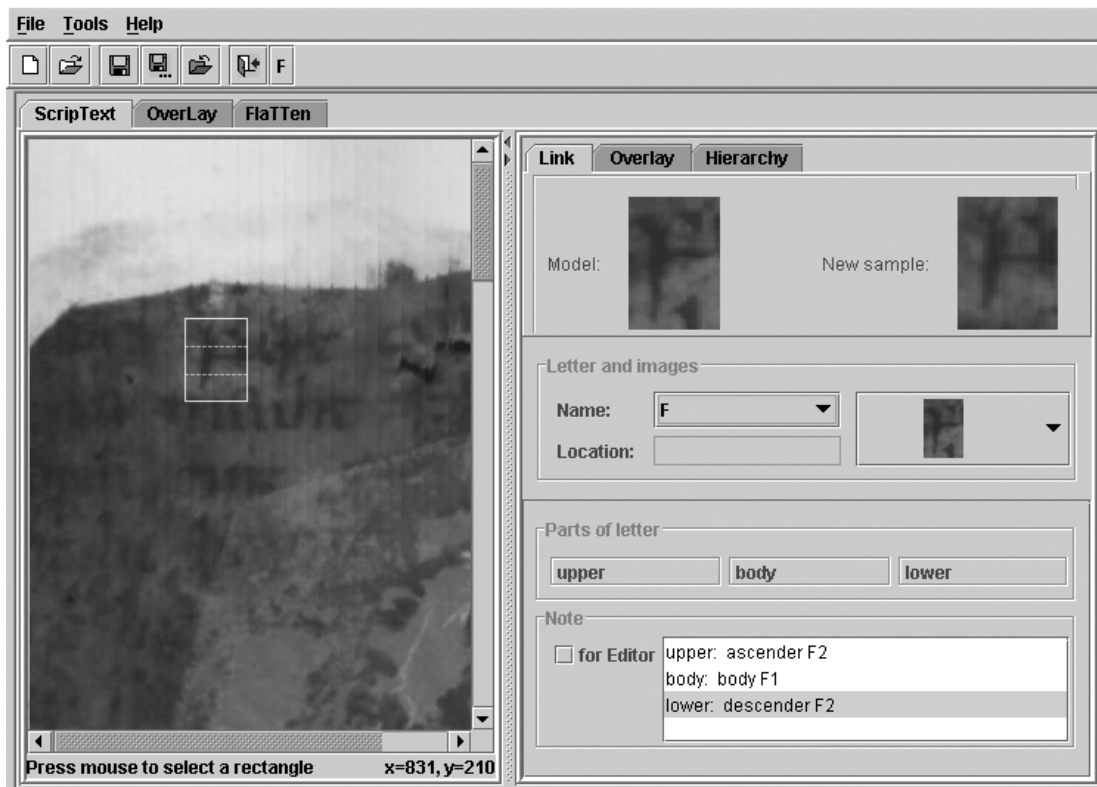


Figure 3: ScripText tool.

The OverLay Tool allows an editor to combine images acquired by different lighting techniques, such as ultraviolet, bright light, and fiber-optic backlighting, minutely compare their different effects on the fly by means of a slide bar that reveals different layers of images, and then tag the transcripts accordingly [fig. 4].<sup>11</sup>

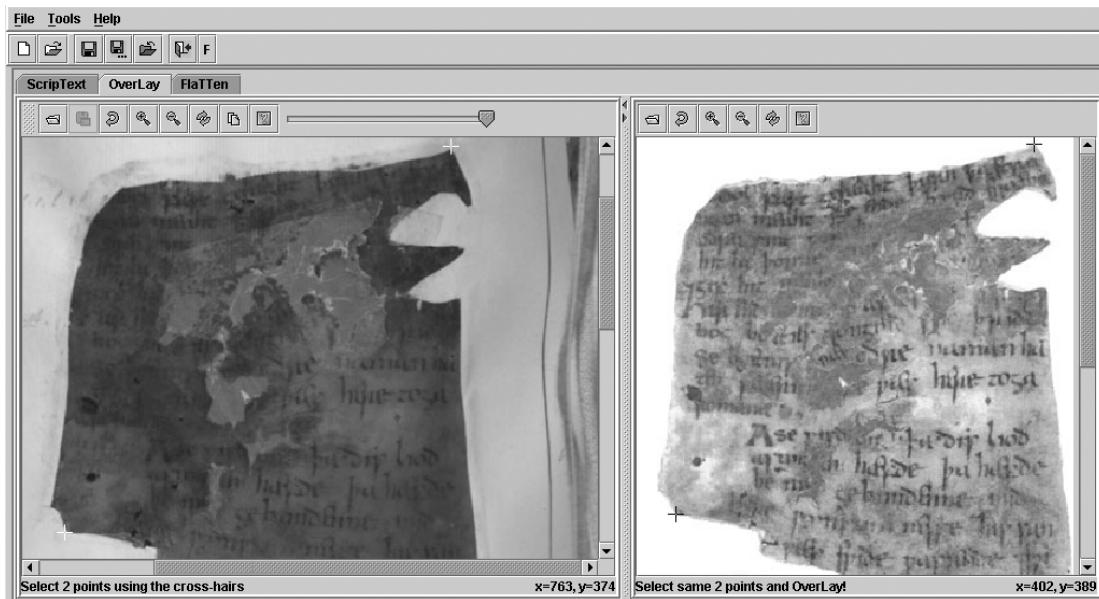


Figure 4: Overlay tool

In the stand-alone version of the search facilities, programming began by attempting to replicate in an open, extensible, XML system the rich functionality of the *Electronic Beowulf* search facilities [fig. 5].<sup>12</sup>

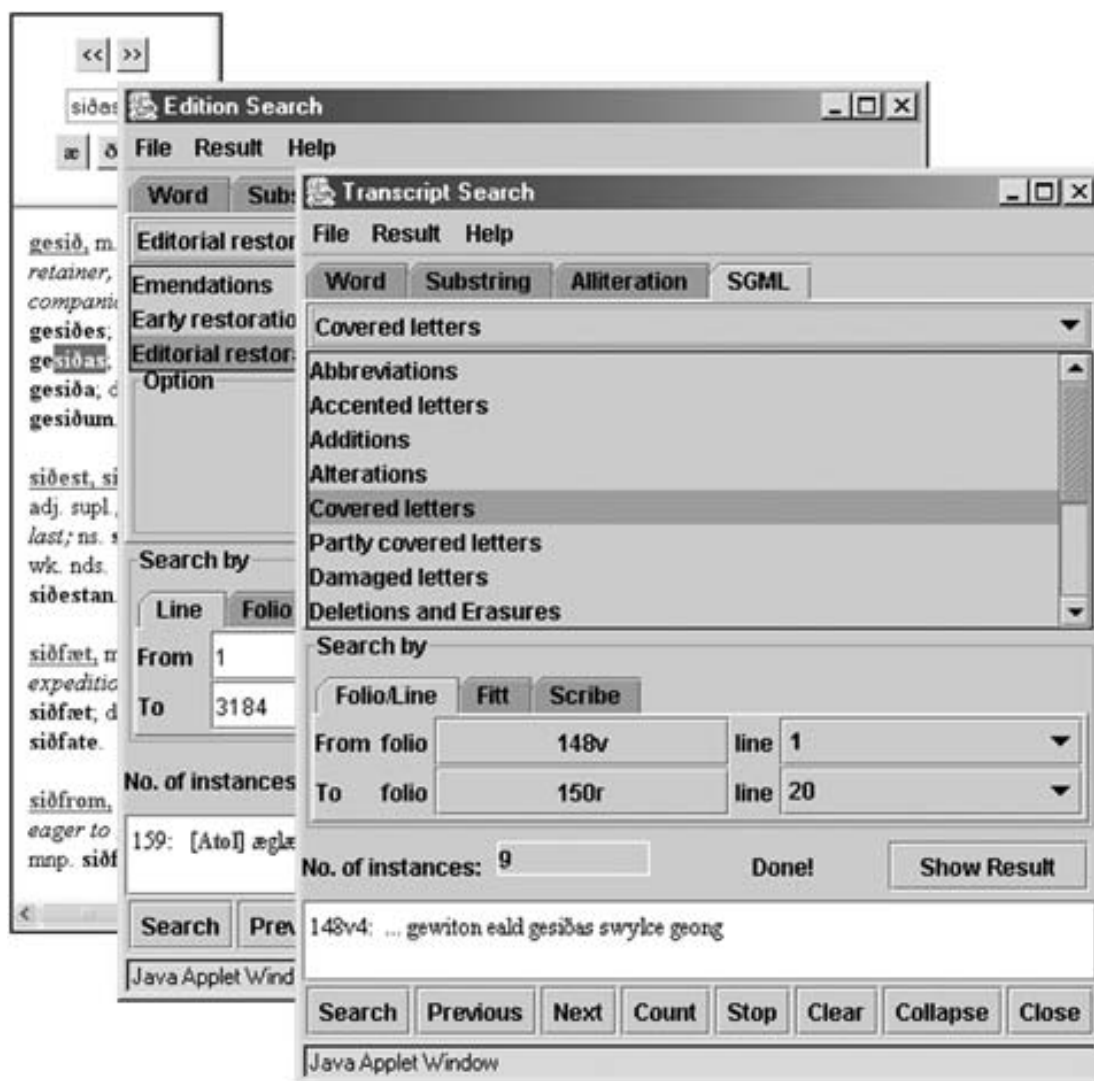


Figure 5: Search tool.

The new version is an extensible approach with an XML interface that will allow a humanities editor to adapt the search facilities to other projects (by changing Fitts to Books, for example, or changing the folio and lines, or the number of scribes) without the need for further programming, simply by changing an XML configuration file.

The stand-alone tools, while performing the editorial functions we needed, force an editor to open files in multiple programs (and remember to save changes and refresh the new file in the other programs), or else to work with the files in one program at a time. A tool developed under the Digital Atheneum project emphasizes the drawbacks of stand-alone approaches. The FlaTTen tool allows editors to remove ambiguous three-dimensional characteristics of an open vellum manuscript further distorted by the restraints of paper frames. By digitally recording the unique 3D shapes of folios, the computer scientists were able to use the geometry of the three-dimensional image as the basis of a "post-warping" and to undo the warping and flatten the image. See Figure 6 for shots of a pre- and post- flattened folio.<sup>13</sup>

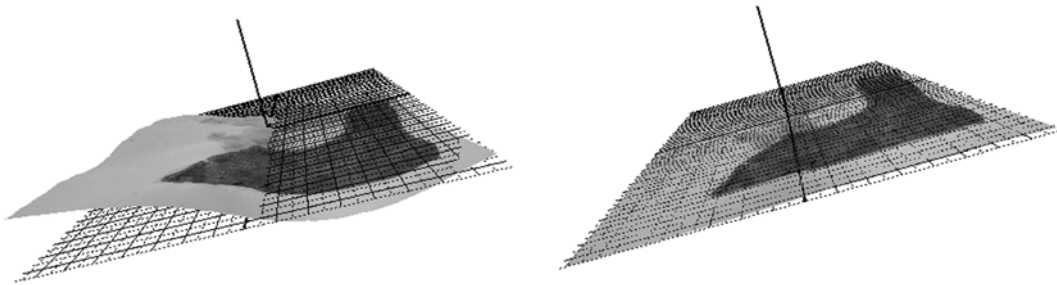


Figure 6: FlaTTen tool.

The FlaTTen Tool is an innovative and sometimes very useful program. For instance, it helped reunite two parts of the same folio that were bound separately during the 19<sup>th</sup> century by digitally "flattening" them, equalizing the distances of the folio fragments in relation to the camera. But the off-the-shelf imaging program that runs the algorithm uses a programming language that is incompatible with our platform, and it would require extensive reprogramming to make a new plug-in. At this point, it seems unlikely that we will be able to integrate this tool into our EPT.

We also had to address the issue of multiple human editors, in our case, the humanities editor and the collaborating research assistants in English and Computer Science. Without some kind of version control, several people working on different aspects of a manuscript, but using the same image files, modifying the same XML files, and updating related software, all lead to chaos in multiple versions and often inadvertent destruction of each other's work. It was apparent that as the project continued we would need to consider ways to avoid editors saving over one another's work. Under the ARCHway project we decided that we needed to bring all these tools, and any others we developed, under a common platform and to develop a DBM (database management) approach for dealing with the concurrent hierarchies inherent in the complex tagging of damaged manuscripts. Another advantage of adopting a common platform was that tools could be developed one-at-a-time, making them suitable projects for Computer Science graduate students, and for truly collaborative development in a teaching and learning environment with the actual humanities users. The tools could be used alone in the platform, but they could also be used in various combinations, communicating with one another through the platform backend. We selected an open-source program as our architectural platform for reasons discussed in detail in 2.3 below. We are still working to recover much of the functionality of the stand-alone tools, but it will be worth it to have the tools integrated under one platform rather acting as completely separate tools.

Users of the completed edition will have access to all these tools to conduct their own research of these collected images. These specialized tools are intended to function together and ultimately to expand the editor's EPT workbench to include a virtual reading room for users of any image-based electronic edition.

## 1.2 THE FINISHED EDITION

The image-based scholarly edition of Alfred's *Boethius* will comprise a complete collection of all manuscript images intricately linked to edition, transcript, glossary, and apparatus files to allow users to view, read, compare, study, and search in an electronic environment that maintains and encourages analysis involving both the edited text and the actual images that establish that particular edition. Editorial glosses, emendations, restorations, punctuation marks, paleographical notes, codicological analyses, and bibliographical materials should all be linked to relevant portions of the manuscript from which the scholarship emerges, taking advantage of the strengths of the electronic medium rather than remaining bound by the typical structure and organization of traditional scholarly editions in printed books. We envision the apparatus of the traditional scholarly edition becoming a pervasively available "Help" whenever and wherever pertinent information relating to the edition, transcript, and glossary files, and their associated images, is required. In this way editorial interventions become completely transparent by the availability of high-resolution images alongside corresponding textual notes, explanatory notes, and bibliographical materials.

While it should provide a variety of views and configurations of the edition, transcript, and image files, the graphical user interface (GUI) for the scholarly edition should also allow users to search the collection exhaustively for words, substrings, grammatical properties, poetic features, and in fact anything that may be of research value or general interest. The search facilities for the *Electronic Boethius* and any other texts under the aegis of the ARCHway Project should eventually be as functional as the ones developed for the *Electronic Beowulf*, but no longer bound to a closed system. Like all the other tools in the EPT workbench, these facilities too will be modular, interoperable, and extensible.

## 2. Architectural Model for ARCHway

The organization of a complex system like the EPT presents a set of serious design and coordination problems. Its workbench must offer comprehensive services, which the editor and co-researchers need to prepare and to maintain DTDs or X-Schemas, encode texts with searchable descriptive and editorial markup in XML, integrate images, and design interfaces. It must also prepare powerful interactive displays for users to take full advantage of the completed image-based edition. Unlike most software projects, whose primary objective is to produce a functional and robust application, ARCHway's goals are more complicated, requiring an infrastructure that integrates Research, Teaching, and Learning among the very different disciplines of computer science and the humanities. Our architectural model must promote interaction between participants and support the development of modular, extensible, interoperable components in a collaborative research and teaching environment that actively involves researchers and students in both disciplines.

### 2.1 RESEARCH, TEACHING, AND LEARNING

The EPT is an innovative system whose strengths hinge on a number of questions that require novel solutions and utilization of emerging technologies. Among these questions are how to effectively, conveniently, and completely map the phenomena of the editorial domain into a software system. The questions lead to further research problems of image archiving, image access and manipulation, methods for automated linking of images with text, storage, access, data persistence and consistency, and virtual remorphing (i.e., virtual repair). The intellectual integrity of the project is being maintained by continuous exchange of "what" and "how" knowledge, and by domain-specific expertise between humanities scholars and computer and technology specialists. Ultimately, the workbench that we are creating will mirror and augment

the processes and methodologies that the editor uses to develop electronic editions from original manuscripts.

The workbench attempts to recreate a virtual environment for studying manuscripts that is reminiscent of the currently prevailing Windows manager interface.<sup>14</sup> As Frederick Brooks (1995, p. 260), an authority in software engineering, puts it, "The WIMP (Windows, Icons, Menus and Pointing) interface is a superb example of a user interface that has conceptual integrity, achieved by the adoption of a familiar mental model, the desktop metaphor, and its careful consistent extension to exploit a computer graphics implementation." However, we envision that the editors' workbench will become more than a metaphor. We aim to build a system that reproduces, augments, and organizes work with manuscript images into image-based editions on the functional, hardware, and almost haptic (i.e., hands-on, touchable) levels. As in any endeavor of developing a complex system, the knowledge, creativity, skills and close collaboration of the people involved are critical to its ultimate success. To accommodate and integrate students at various levels of their studies and researchers from interdisciplinary environments the architecture must be flexible and open for creative solutions and must respond well to potential errors, and creative changes, which are unavoidable in any scholarly process.

## 2.2 SOFTWARE ARCHITECTURE FOR THE EPT

The EPT uses a variant of the skeletal system advocated by Harlan Mills (1971). In this approach the EPT grows from a few basic plug-ins, which implement the core functionality. These plug-ins also provide extension points that allow for further development by adding functionality and by gradual integration of new components. Furthermore, to expedite the deployment of usable tools, we began by scheduling the work to incorporate the tools that the humanities scholars have developed in

prototype, which they need in the early stages of the editing process, and then add new plug-ins as they are needed.

The most advanced researchers and programmers are developing the skeleton that comprises the data source layer and the navigator that ties together all the tools and provides a convenient Application Programming Interface (API) for the tool developers. This framework helps maintain the accuracy of the system and at the same time separates and distributes design and implementation tasks among different groups of students. The encapsulation and implementation of plug-ins allow developers to define self-contained modules such as parsers, search facilities, and image manipulation tools, while limiting interaction between separate components to interfaces, thus reducing the probability of errors. These modules are plugged into the skeletal system one by one as they are developed, becoming available immediately to the user. The users then have the ability to create their own suites of tools most adequate for the current task. The collection of plug-ins is modeled as a single workbench, yet each plug-in can be viewed and used individually, at the same time sharing a unified graphical interface and unified access to data with other plug-ins in the workbench. With an easily configurable GUI (Graphical User Interface), editors will have ready access to different tools and will be able to customize the appearance and functionality to fit their needs and work style.

### 2.3 IMPLEMENTATION OF ARCHITECTURE

We successfully tested the plug-in approach in a prototype for XML searches using an eXist database (Turner 2002). For the production development we have adopted Eclipse, an open platform for tool integration originally built by IBM and then released to the open source community, which continues to enhance and maintain it. As described in the Eclipse White Paper,<sup>15</sup> Eclipse facilitates the development of new tools, supports GUI rendering, and supports both XML and images and the seamless

integration of tools built by independent teams. It then allows for the deployment of these tools on a wide range of operating systems including Windows, Linux, and to some extent, Macintosh, whose current operating system does not work well with Java SWT and standard browsers.<sup>16</sup> Eclipse attracts an active worldwide developer community and has strong support from industry-leading companies.<sup>17</sup> For the humanities computing community, it is notable that there are already relevant tools available for Eclipse including `<oxygen/>`, a TEI-compliant XML editor which has been released as an Eclipse plug-in. This choice of an existing and well-documented platform allows us to use our limited resources in an efficient way and focus the development on the tools for the editors' workbench. The alternative would be to develop our own, similar platform, but substantial effort and time would be needed, as well as the focused resources of a software company, rather than a research project.

Our implementation of the EPT has three layers, the first being the functionality provided by Eclipse. The second layer contains the Data Source Layer and the Project Explorer that have been built on top of Eclipse to extend the functionality to better serve our needs. The third, topmost layer contains tools for both editor and user. All these layers contain Utility plug-ins that extend the functionality or provide features that span multiple tools. Figure 7 provides a schematic view of the EPT architecture.

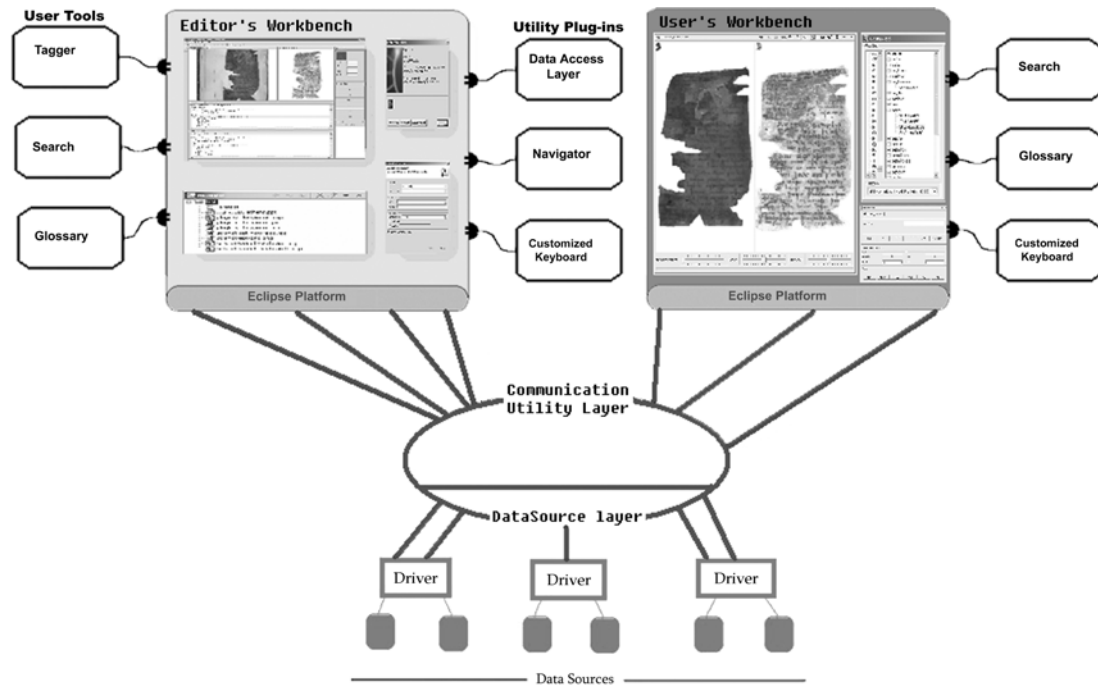


Figure 7: Schematic view of the EPT architecture

The Data Source Layer defines generic data-source independent APIs that provide transparent access to data from the file-system, network, and various databases. This layer allows the user to import/export projects between these data sources. Its design insulates the plug-ins from the many different types of sources that store the data (images and XML files), and permits the addition of new data sources without affecting the existing functionality.

The Project Explorer organizes all the electronic editions available to the user from various data sources and presents their logical view in the form of a single browseable and expandable tree. The user can access and open any of the projects and resources with the tool or plug-in of their choice by clicking on the project where it occurs in the project navigation tree. The user can associate a list of tools and actions with the nodes of the tree; the platform can automatically launch a default

action or the user can select a tool or action from the list. Tools can set decorations, or *flags*, marking the status of each element of the project (a reminder, for example, to save a modified resource).

The Utilities plug-ins provide common functionality, and are used by multiple tools. The Keyboard plug-in is an example: it provides toolbars representing special characters in a given alphabet (for example the Old English letters æ, ð, and þ), which the user can enter into search-fields, glossary-fields, and into the edition xml files. Since the keyboard is defined as a single, simple XML file, humanities editors can easily define keyboards as needed, without the assistance of a programmer. Simple XML-based descriptors and configuration files are used for other features of the EPT as well.

The topmost layer contains the actual tools for the editors and users of the electronic editions. In general, each tool is implemented as a plug-in. APIs available from the supporting layers facilitate development of the tools and code reuse, and allow the developer to combine multiple plug-ins into more comprehensive tools. As described in 1.1 above, the *Electronic Boethius* and the Digital Atheneum projects developed many prototype stand-alone tools, which with the fully functional ones from the *Electronic Beowulf* serve the programmers as indispensable models. However, as they exist in a closed, non-extensible system, they now require reprogramming. We are now working to implement their functionality in Eclipse plug-ins, and see immediate visual enhancement from using the new uniform framework and flexible GUIs. For example, the Eclipse's view and editor window concepts allow the programmers to effortlessly separate components of the Search tool, so users can individually close them and save valuable screen space while still retaining the search results. Another noticeable advantage is that the Java SWT (Standard Widget Toolkit) that Eclipse implemented based on system specific GUI, all the tools not

only automatically receive a native look on Windows, Mac and Linux platforms, but also perform faster than Java Swing widgets. In short, we are convinced that modifying and integrating the stand-alone tools into the Eclipse framework will create a system far more beneficial to the editor than designing the EPT as several stand-alone programs, or as one single large program.

Our experience shows that the proposed architecture for EPT and its mapping into Eclipse advances the development of image-based electronic editions, provides flexibility for research, actively engages students and enhances their learning process, while at the same time enforcing a rigorous software development. In a broader context, by embracing the proposed architecture and the Eclipse platform, any member of the humanities computing community can contribute specialized tools to the EPT workbench and thus build a more far-reaching collaborative project.

### **3. Management of Markup**

The process of preparing electronic editions is long, time-consuming, and arduous for an editor or editors.<sup>18</sup> Some of the work cannot be automated -- editors, for example, must scrutinize every single letter of a document numerous times to come to fully informed decisions concerning script, meaning, and spelling.<sup>19</sup> Correct manipulation of the data that forms the electronic edition lies at the core of the success of the EPT. However, the problem of management of the XML encoding of the manuscripts is made more difficult by the complexity of the markup language. Management of XML data must be both convenient for the editors and users of the editions *and* efficient. To this end, it is essential to implement the following tasks:

*Generation of markup.* The actual process of furnishing markup lies at the heart of the human-computer interaction in creating image-based electronic editions. The human editor makes an encoding decision and then records it using the interface of

the XML editing software, for example, by highlighting the part of the texts that needs to be encoded and selecting the XML element to insert.

*Storage of Markup.* All markup the editors generate must be preserved in some form of repository, which may be a plain XML file, a collection of XML files, or a database.

*Query Processing.* The ability to query encoded information is the principal reason for building electronic editions using XML.. Thus, the designers must fully understand the information needs of editors and users of electronic editions to provide adequate query-processing support for them.

*XML Document Generation.* Although there is no special reason for editors and end-users of electronic editions to work directly with the actual codes in XML documents, editors must nonetheless generate valid and well-formed XML documents to ensure, among other things, proper preservation of the data comprising the electronic edition.

In most current approaches to editing of XML documents, the four tasks outlined above (or some combination of them) are typically merged. For example, a typical XML editor allows the user to introduce markup, immediately preserves the markup in an XML document (solving thus the task of storage of markup and generation of the XML document) and uses the XML document as a base for searching. However, as the complexity of XML markup (and with it, of queries) and the size of the document grow, separation of these tasks becomes the key to an efficient and convenient data management strategy.

The XML management framework we propose for the ARCHway project is designed not only to treat these tasks as separate, but also as maximally *independent* from each other. This framework gives us the ability to adopt flexible and efficient solutions for

each individual task. Figure 8 below illustrates the general approach that we have adopted. A special-purpose database management system is used for storage and retrieval of document-centric XML. The database is entrusted with the storage of all the encoding information that comes from the editor of the electronic edition. The query processing module will accept search queries from other components of the system, execute them, and return results. The data stored in the database can also be used to generate actual XML documents.

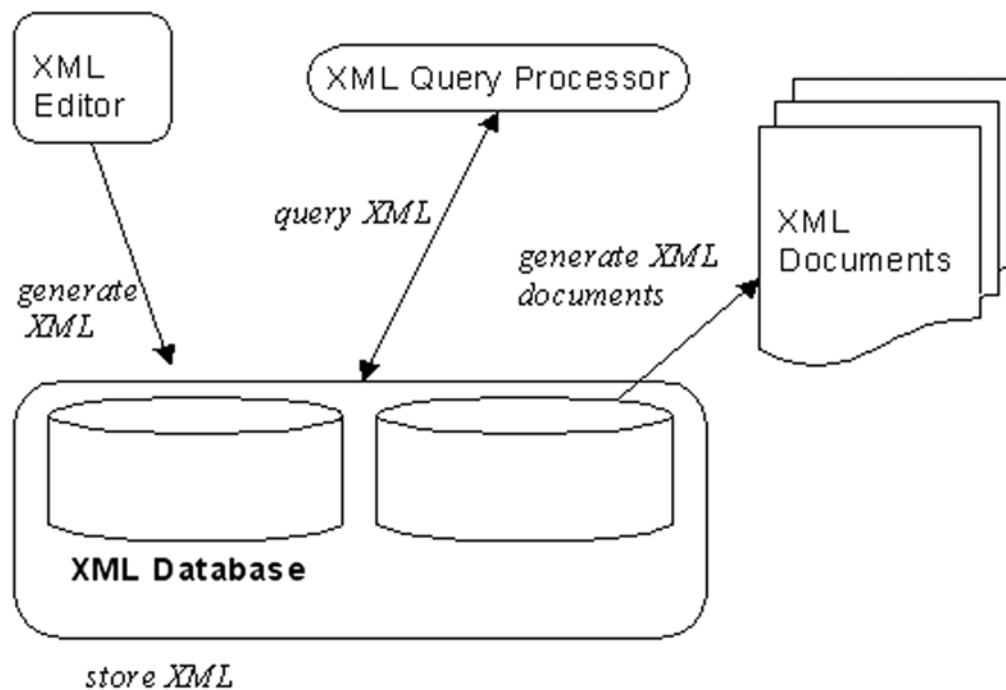


Figure 8: The ARCHway approach to management of XML.

### 3.1 DEALING WITH CONCURRENT XML HIERARCHIES

The proposed XML-document framework alleviates the problems associated with the creation and maintenance of complex XML markup. The problem of XML markup with overlapping scope is known in the XML processing community as the problem of *concurrent XML hierarchies*. It often occurs in document-centric XML when two or more markup elements come from different *logical* hierarchies and therefore may result in non-well-formed XML. Consider, for example, two lines from British Library MS Cotton Otho A. vi (folio 38 verso, lines 9-10):

```
Se Boetius wæs odre naman ha
ten Seuerinus se wæs heretoga
```

There are at least two sets of overlapping features in this fragment. One is between the physical line markup and word markup, as the single word, *haten*, starts on line 9 and ends on line 10. In addition, there is a hole in the manuscript that covers parts of the words *Seuerinus* and *se*. Using just the word `<w>`, line `<line>`, and damage `<dmg>` elements, we would find it easy to encode this fragment as:

```
<line> ... <w>ha</line>
<line>ten</w><w>Seuerin<dmg>us</w><w>s</dmg>e</w>... </line>.
```

However, this encoding is not well-formed.

Past approaches for dealing with concurrent hierarchies have taken paths unsupported by XML, such as using the SGML CONCUR function; or using non-XML markup languages, such as TexMECS (Huitfeldt et. a. 2001); or resolving markup conflicts by manipulation of the DTD. For example, the TEI Guidelines propose a variety of *ad hoc* encoding tricks to resolve the most common overlap conflicts.<sup>20</sup> Among the suggestions are the use of milestone elements, instead of elements with scope, and fragmentation of elements. For example, making `<line>` a milestone (empty) element resolves the conflict of the `<line>` and `<w>` discussed above:

```
<line/> ... <w> ha
<line/>ten</w> ...
```

Using fragmentation, another TEI Guidelines suggestion, we can break the damage tag into two parts and connect them using special attributes `next` and `prev`:

```
...<w>Seurin<dmg id="1" next="2">us</dmg></w>
<w> <dmg id="2" prev="1"> s </dmg> e </w>...
```

The use of such solutions requires human editors to make the right decisions both at the time of creation of the markup language (identification of milestone elements, elements that can be fragmented, all possible element overlaps), and at the time of markup. The DTD for the markup language becomes cluttered with rules specifying all possible overlaps, and is hard to maintain. The quality of the editorial process also suffers: different tags leading to overlapping markup can be introduced at different stages of editing, and, possibly, by different editors. This makes the use of fragmentation very inconvenient. When introducing new markup, the editor must manually go through the encoded content, determine all markup that overlaps with the scope of the new tag, and introduce correct fragmentation. At the same time, use of milestone elements in lieu of elements with scope makes it harder to process queries that rely on “virtual scope” of such tags. As a result, the query-processing becomes excessively dependent on the inherent restrictions of a DTD.

Durusau et. al. (2002) propose a new approach to dealing with concurrent XML. They identify the actual concurrent hierarchies, subsets of the markup language that form a single hierarchy and do not overlap, and construct a DTD for each such hierarchy.<sup>21</sup> After that, they use Xpath expressions to encode each word of the content with the tags from each DTD. Their method is flexible and expandable. It is easy to create a new hierarchy and add markup to the document based on it, but the XML produced this way is difficult to read, because it contains complex XPath references to the DTDs that form the hierarchy.

In Dekhtyar et. al. (forthcoming), we have introduced the foundations for our approach to dealing with concurrent hierarchies. Following Durusau et. al. (2002), we have defined a notion of concurrent DTD: a collection of DTDs that share the root element but no other elements. We then defined a notion of a concurrent XML document as a collection of encodings of the same content in the DTDs that form the concurrent DTD. We have developed the algorithm for converting a concurrent XML document into a single XML document in unified markup using a version of TEI's fragmentation technique to *automatically* and efficiently deal with element overlap.

The theoretical framework of Dekhtyar et. al. (forthcoming) is implemented in ARCHWay as follows. We notice that concurrent markup poses a problem only at the stage of XML document generation. The markup database of a document can store information about overlapping markup and use it to process queries without a problem. Our principle of task separation allows the editor *not to worry* about concurrent markup representation during markup creation stages. Information goes to the database where it is duly stored. At the same time, we construct the “ultimate” XML document containing the entire encoding in a variety of ways, using different techniques proposed by TEI Guidelines and/or Durusau et. al. (2002), among others. Because information stored in the database is sufficient to produce any such encoding, we can implement a series of XML generation drivers for the desired techniques and leave the choice of the specific representation to the editor. At the same time, search queries will rely on the content of the database, rather than on the specifics of syntax of the “ultimate” XML document, and query processing algorithms are, as a result, independent of the XML syntax.

### 3.2 COMBINING XML MANAGEMENT TASKS

There are some significant remaining issues associated with each of the four XML management tasks listed above. For XML generation, we must design and develop user interfaces that editors and users of electronic editions will find convenient and effective. For XML storage, we must design and implement efficient mechanisms for storing document-centric XML in a relational database. Related to this issue is the problem of choosing the right query language for the XML data and implementing efficient query processing. While the choice of a specific XML query language, such as XQuery (Boag et. al. 2003) need not depend on the choice of the XML data storage strategy, the implementation of the query processor certainly does depend on this choice. Finally, as mentioned above, we must anticipate that editors will choose a variety of ways to represent complex encoding in a real XML document.

With the possible exception of aspects related to storage and querying of XML data, our chosen framework allows us to address these questions independently. The advantage of our approach is that change in one part of the XML management will not affect other parts. For example, as Tian (2002) showed, although there exists a variety of methods for converting XML encodings into relational databases, none of them is obviously better than all others. As experiments are run to establish the XML-to-relational database conversion most appropriate for our application, there is no need to change the XML editor GUI, query language used, or the format of the output XML documents (we must, however, implement query processing and XML document generation for each approach separately). Similarly, introduction of a new format for output XML, leads only to development of appropriate XML generation code based on the database content, but does not affect how XML is stored, or how searches are performed.

## **Looking Ahead**

In this paper we have introduced the concept of the Edition Production Technology, sketched a history of its development, and discussed in detail the architecture and data management aspects of the project. We have proposed an open-source Java-based Eclipse as the development and integration environment for the EPT workbench. We have discussed the EPT as it relates to the *Electronic Boethius*, but our intention is that, when completed, the EPT will be useful for many other image-based humanities computing projects.

The most attractive feature of the EPT is its extensibility, rendering it adaptable to any project requiring detailed markup and the integration of text and image. Graphical user interfaces for the tools are automatically customized based on the DTD, and in some cases additional information is taken from an XML configuration file. Editors may assign alternate names within the DTD for tags to be shown on the GUI buttons (“damage” instead of “dmg”). Any editor can easily modify the glossary tool using XML configuration files to suit different languages. Finally, the resulting edition will have full search capabilities, because the GUI for the search tool is also built from the DTD, XML markup, and XML configuration files.

Since the tools are designed to draw the information for their GUIs from the collections of images, the DTDs, XML configuration files, and XML encoded documents, we anticipate that no software development experience will be required for editors to use the EPT for projects other than the development of image-based electronic editions of Old English manuscript.

## Notes

<sup>1</sup> This article is based on work supported by the National Science Foundation under Grant No. 0219924, awarded pursuant to the authority of the NSF Act of 1950 (42 U.S.C. 1861 et seq.). It is subject to GC-1 Grant General Conditions (10/98) and is made in accordance with the provisions of NSF 98-63, "Information Technology Research."

---

<sup>2</sup> The *Electronic Boethius* Project is funded by a Collaborative Research Award from the National Endowment for the Humanities and the Andrew W. Mellon Foundation, and is sponsored by The British Library and the Bodleian Library, Oxford, who are providing digital images of the relevant documents.

<sup>3</sup> A. Dekhtyar and I. E. Jacob, "A Framework for Management of Concurrent XML Markup," International Workshop on XML Schema and Data Management (XSDM'03), forthcoming in *LNCS* (Springer-Verlag).

<sup>4</sup> For a discussion of the Ashburnham House fire of 1731 and modern prospects for the preservation and restoration of the damaged manuscripts, see A. Prescott (1997).

<sup>5</sup> For a fuller description of the relationships among the surviving manuscripts and previous editions, see K. S. Kiernan (1998).

<sup>6</sup> Other image collections we have available are from British Library MS Cotton Vitellius A. xv (the *Beowulf* manuscript), Cotton Otho B. x, and the Boethian manuscripts at Oxford University, Bodleian Library MSS Bodley 180 and Junius 12.

<sup>7</sup> Digital Atheneum Project: New techniques for restoring, searching, and editing humanities collections, sponsored by the National Science Foundation's Digital Libraries Initiative Phase 2. <http://www.digitalatheneum.org>. For more on the work of the Digital Atheneum Project see Seales (1999).

<sup>8</sup> Emil Iacob programmed the Glossary Tool as part of a larger program called the Electronic Editions Editor (E3), which included the GT, Tagger, and an XML validator and transformer.

<sup>9</sup> Anshul Jain began reprogramming the search facilities (originally programmed by C. J. Yuan) of the *Electronic Beowulf* by Kiernan et al. (1999) for generic use as part of the *Electronic Boethius* project.

<sup>10</sup> This work was programmed by Chengdong Li in consultation and collaboration with Emil Iacob.

<sup>11</sup> This work was programmed by Chengdong Li.

<sup>12</sup> For a full description of the *Electronic Beowulf* search facilities, go to the online Guide at <http://www.uky.edu/~kiernan/eBeowulf/main.htm> and follow the link from the Help Index to Search Facilities.

<sup>13</sup> For more about the development and uses of the FlaTTen Tool see Brown and Seales (2001) and Kiernan et. al. (2002).

<sup>14</sup> Douglas Engelbart, director of the Augmentation Research Center at Stanford University in the late 1960s, conceived and developed the oN-Line System (NLS), the first implementation of the visual concept of "windows" in computers.

<sup>15</sup> Object Technology International, Inc. (2003).

---

<sup>16</sup> SWT implementation for Macintosh started later than for the other platforms, and the Macintosh release of Eclipse 2.1 was still not mature at the time of its release. Observable problems included difficulties with changing the arrangement of elements in the Eclipse display with a mouse, and the mis-rendering of some fonts. There is currently intensive Mac development of Eclipse, and it is anticipated that future releases will fix these problems. The most recent release of the Eclipse Platform is designed to run on Windows 98/ME/2000/XP, Red Hat Linux Version 7.1 (x86/Motif and x86/GTK), SuSE Linux 7.1 (x86/Motif and x86/GTK), Solaris 8 (SPARC/Motif), QNX (x86/Photon), AIX (PPC/Motif), HP-UX (HP9000/Motif), and Mac OSX (Mac/Carbon).

<sup>17</sup> Eclipse board member organizations include IBM, Red Hat, Hitachi, Scapa Technologies, Hewlett-Packard, Fujitsu, and Fraunhofer, among many others (see <http://www.eclipse.org/org/index.html> for more information on the Eclipse consortium).

<sup>18</sup> Seales et. al. (2000)

<sup>19</sup> Hayes (2001)

<sup>20</sup> For a full discussion of the ways that the TEI Guidelines approach the problem of overlapping elements, see Sperberg-McQueen and Burnard (2001), Chapter 31 “Multiple Hierarchies.”

<sup>21</sup> For example, the *Electronic Boethius* project has constructed separate hierarchies for paleographical, codicological, manuscript condition, editorial, and textual markup (among others).

## References

### Primary Sources

British Library MS Cotton Otho A. vi.

Oxford Bodleian Library MS Bodley 180.

Oxford Bodleian Library MS Junius 12.

### Editions

Krapp G. P., ed. (1932) *The Paris Psalter and the Meters of Boethius*. The Anglo-Saxon Poetic Records 5. Columbia University Press, New York.

---

Robinson F. C., Stanley E. G., eds. (1991) *Old English Verse Texts from Many Sources: A Comprehensive Collection*. Early English Manuscripts in Facsimile 23. Rosenkilde and Bagger, Copenhagen, Denmark.

Sedgefield W. J., ed. (1899) *King Alfred's Anglo-Saxon Version of Boethius, de Consolatione Philosophiae*. Clarendon Press, Oxford.

#### Secondary Sources

Adler, S., et. al. (2001) Extensible Stylesheet Language (XSL) Version 1.0. W3C Recommendation, 15 October 2001.

Biron P. V, Malhotra A. (2001) XML Schema Part 2: Datatypes. W3C Recommendation, 2 May 2001.

Boag S., et. al. (2003) XQuery 1.0: An XML Query Language. W3C Recommendation, 2 May 2003.

Bray T., Paoli J., Sperberg-McQueen C. M., Maler E. (2000) Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, 6 October 2000.

Brooks F. P., Jr. (1995) *The Mythical Man-Month: Essays on Software Engineering*. Wesley Longman, Reading, Mass.

Brown M. S., Seales W. B., Kiernan K. S., Griffioen J. (2001) 3D Acquisition and Restoration of Medieval Manuscripts. *Communications of the ACM: Special Issue on Digital Libraries*, 44/5 May 2001, p. 58.

Brown M. S., Seales W. B. (2001) The Digital Atheneum: New Approaches for Preserving, Restoring, and Analyzing Damaged Manuscripts. *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM Press, New York, pp. 437-443.

- 
- Dekhtyar A., Jacob I. E. (Forthcoming) A Framework for Management of Concurrent XML Markup. International Workshop on XML Schema and Data Management (XSDM'03), forthcoming in *LNCS* (Springer-Verlag).
- Durusau P., O'Donnell M.B. (2002) Concurrent Markup for XML Documents. In *Proceedings of XML Europe*, May 2002.
- Eclipse Platform, 7 July 2003 [<http://www.eclipse.org>]
- Gamma E., Helm R., Johnson R., Vlissides J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Mass.
- Goodman J. E., O'Rourke J. (1997) *Handbook of Discrete and Computational Geometry*. CRC Press, Boca Raton.
- Hayes D. (2001) Glossing Damaged Manuscripts: an Example from Ælfric's Lives of Saints. *Digital Resources for the Humanities* (DRH01). University of London, London, UK. 10 July 2001.
- Huitfeldt C., Sperberg-McQueen C.M. (2001) TexMECS: An experimental markup meta-language for complex documents. February 2001 [<http://www.hit.uib.no/claus/mlod/papers/texmecs.html>]
- Hunt A., Thomas D., Cunningham W. (1999) *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley, Reading, Mass.
- Kiernan K. S. (1998) Alfred the Great's Burnt Boethius. In George Bornstein and Theresa Tinkle (eds.), *The Iconic Page in Manuscript, Print, and Digital Culture*, University of Michigan Press, Ann Arbor, pp. 7-32.
- Kiernan K. S., Prescott A., French D., Solopova E., Cantara L., Ellis M., Yuan C. J. (1999) *Electronic Beowulf*. 2 CD-ROMS. British Library Publications, London and University of Michigan Press, Ann Arbor.

- 
- Kiernan K. S., Seales B., Griffioen J. (2002) The Reappearances of St. Basil the Great in British Library MS Cotton Otho B. x. *Computers and the Humanities* 36/1, February 2002, pp. 7-26.
- Mills H. D. (1971) Top-down Programming in Large Systems. In R. Rustin (ed.), *Debugging Techniques in Large Systems*. Prentice-Hall, Englewood Cliffs, NJ.
- Object Technology International, Inc. (2003) Eclipse Platform Technical Overview. February 2003. Originally published July 2001. [<http://www.eclipse.org/whitepapers/eclipse-overview.pdf/>.]
- Prescott A. (1997) 'Their Present Miserable State of Cremation': The Restoration of the Cotton Library. In C. J. Wright (ed.), *Sir Robert Cotton as Collector: Essays on an Early Stuart Courtier and His Legacy*. British Library Publications, London, 391-454.
- Seales W. B., Griffioen J., Kiernan K. S. (1999) The Digital Atheneum - Restoring Damaged Manuscripts. *RLG DigiNews* 3/6, 15 December 1999. [<http://www.rlg.org/preserv/diginews/diginews3-6.html#technical1>]
- Seales W. B., Griffioen J., Kiernan K. S., Yuan C. J., Cantara L. (2000) The Digital Atheneum: New Technologies for Restoring and Preserving Old Documents. *Computers in Libraries* 20/2, February 2000, pp. 26-30. [<http://www.infoday.com/cilmag/feb00/seales.htm>]
- Sperberg-McQueen C. M., Burnard L. (2001) *Guidelines for Text Encoding and Interchange* (P4), The TEI Consortium.
- Tian F., DeWitt D.J., Chen J., Zhang C. The Design and Performance Evaluation of Alternative XML Storage Strategies. *SIGMOD Record*, 31/1, March 2002.
- Turner J. A. (2002) *Customized Graphical XML Search Tool*, Master's Project, Department of Computer Science, University of Kentucky, December 2002.

- 
- World Wide Web Consortium (W3C) (2001) "The Extensible Stylesheet Language (XSL).  
[[http://www.w3.org/Style/XSL/Overview-table.html/.](http://www.w3.org/Style/XSL/Overview-table.html/)] Last modified 10 July 2001.
- Yuan C. J., Seales W. B. (2001) Guided Linking: Efficiently Making Image-to-Transcript Correspondence. *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM Press, New York, p. 471.