The TEIViewer: Facilitating the transition from XML to web display

Stephanie A. Schlitz
Department of English, Bloomsburg University of Pennsylvania, Bloomsburg, PA, USA

Garrick S. Bodine
Web Projects and Applications, Penn State University, University Park, PA, USA

Abstract

This article addresses the need for TEI display tools. In order to illustrate the need for display tools, we begin with a brief review of the tools that are currently available, summarizing in particular those listed on the TEI Wiki Tools page. We then turn to a discussion of our work on the development of the TEIViewer (http://teiviewer.org), a simple, JavaScript-driven, portable display tool designed to facilitate the online representation of and interaction with elements and attributes described within select modules of the TEI P5 Guidelines and encoded as layers of data and metadata in TEI-XML documents. We explain how the TEIViewer works by describing the interactions between the XML source layer, the display layer generated via XSL, and the interactive layer powered by jQuery and CSS; and we explain why we chose the jQuery JavaScript library to manage the Viewer's functionality as well as the advantages of this decision. Finally we describe current implementations and plans for release.

Correspondence:
Stephanie A. Schlitz, 117 B Bakeless Hall, Bloomsburg, PA 17815, USA.
E-mail:
sschlitz@bloomu.edu

1 Introduction

> Tools for creating, editing, transforming, and publishing TEI documents and schemas are an essential part of using the TEI Guidelines. (TEI: The Text Encoding Initiative, 2007d)

Encoding texts in Extensible Markup Language (XML) according to the Text Encoding Initiative (TEI) Guidelines often represents only a single, albeit it critical and complex, stage within the lifecycle of a digital project. The ultimate goal for many text editors and encoding projects is delivery: to make their encoded documents available for research and teaching purposes. Very often this requires that their XML source documents be transformed and displayed on the Web. Although the TEI XSL stylesheet library developed by Sebastian Rahtz and maintained by the TEI may be the first and most obvious resource for those who wish to publish or display their work, few additional Web delivery tools are available. Perhaps for this reason, TEI users continue to express an interest in the development of display tools. At the 2003 TEI Members Meeting, for instance, the consensus of the Tools Special Interest Group (SIG) 'was an interest in web delivery tools' (2007a). Similar interests were expressed at the TEI Members Meeting in 2007, when again the Tools SIG noted an interest in 'simple' display tools and in addressing the need to provide information about publishing TEI documents on the Internet (2007c).

Literary and Linguistic Computing, Vol. 24, No. 3, 2009.

doi:10.1093/llc/fqp022 Advance Access published on 14 May 2009

The TEI Tools Wiki page, a resource which is maintained by the TEI, lists 15[1] tools in the 'Publishing and delivery tools' category (2008). Although at first glance such an extensive list may lead to the impression that TEI users have a range of publishing and delivery options, a closer examination illustrates that most of these tools fall into one or the other of two[2] general but distinct categories. The first of these categories, publishing and delivery tools in the broadest sense, describes tools which are fundamental to various aspects of the building of finished delivery applications. Such a category can—and does—include a broad swath of programming languages, markup tools, development frameworks, and backend applications for text-analysis and database storage and retrieval, all designed for users of widely varying skill levels. Notably, however, these kinds of applications do not result in Web delivery without the support of additional applications and/or subsequent, and often complex, steps.

The majority of the 15 tools listed on the 'Publishing and delivery tools' page fall into this first category. Thus, databases such as Berkeley DB XML, eXist, MarkLogic Server, and RefDB; database search and retrieval systems such as PhiloLogic and Xaira; web development frameworks such as Apache Cocoon; library 'access' tools such as DLXS; image annotators such as Image Markup Tool; programming languages such as XSLT and XQuery; and corpus annotators such as the Sacodeyl Annotator, all fall within the broader category of publishing and delivery tools. Although individuals and projects can use these applications and programming languages to facilitate various kinds of publication and delivery, such applications do not, in and of themselves, allow for the simple and easy publication of TEI encoded documents on the Web.

The second category, publishing and delivery tools in the narrower sense, generally describes tools which have been designed to display documents in human-readable form on the Web yet require little or no programming expertise to accomplish this end. Only three entries, the <teiPublisher>, 'an extensible, modular and configurable xml-based repository' (2004); the Classical Text Editor, a tool which converts word processing or XML files to various types of files for printing or Internet publication (Hagel, 1997); and TEI-P5-XSL, a reference to Sebastian Rahtz's stylesheet library (which is linked elsewhere within the XSLT entry), fall into this category. However, only Rahtz's stylesheet library and the Classical Text Editor are currently maintained. While the <teiPublisher> appears at one time to have been an active and promising project, its SourceForge download and forum data suggest that today the tool is all but moribund.

The larger 'Tools' and 'Analysis tools' categories include an additional publishing tool, the Versioning Machine. Originally released in 2003, this application provides 'a framework and an interface for displaying multiple versions of text encoded according to the Text Encoding Initiative (TEI) Guidelines' (Schreibman *et al.*, 2003). Although this tool does require some configuration by users, its display framework and comparing capabilities are already complete. While the value of this tool, particularly in its contribution to discussions of tool development in general, cannot be overstated, it has not been employed by a wide berth of users (to date, there has been one implementation), perhaps because it was designed to meet the needs of a narrowly defined audience (Schreibman *et al.*, 2007).

In short, only three of the 15 tools described above fall into the category of narrowly defined publishing and delivery tools, and this in part explains why an interest in the development of additional display tools persists. Moreover, we believe there is a need for tools designed for individual TEI users, those with little or no technical support, those with limited resources, beginners, and even potential TEI users, who otherwise will continue to find themselves with encoded documents but with few resources to support the next stage in the life-cycle of their digital projects or, in the case of prospective users, with little incentive to begin their projects in the first place.

2 The TEIViewer

Our decision to enter into the area of tool development stemmed from our own need to identify the best means for displaying our work. Because

we desired a browser-based representation that would (1) enhance the human readability of our documents and (2) allow end-users to interact with the layers of encoded metadata, and because we could not find an existing tool to meet these needs, we began planning our own solution. In doing so, we faced a number of challenges. Identifying a systematic approach to the decision making process was among the foremost. In order to ensure that all vital points of view were represented as we continued into the design and development stages, we outlined three fundamental perspectives to direct our work:

(1) Content expert and encoder: knowledge of TEI modules, including elements and attributes defined within the project's schema; responsible for encoding decisions and for documenting and communicating these to the programmer.
(2) Programmer: interpret encoder's decisions; interpret requests; recommend methodologies; implement methodologies.
(3) End-users: readers, researchers, and teachers who utilize and manipulate texts for various types of analysis.

We also turned to the work of Edward Tufte (2001), adopting a standard design principle that we believed echoed the needs of the TEI community: 'Presenting large amounts of information in a way that is compact, accurate, adequate for the purpose, and easy to understand' (UW Technology, 1999). In striving to meet this design goal in the context of textual documents, we chose a three-level approach based on the common web development and programming tenet of separation of textual content from display and interactivity features (see, for example, Cohen, 2004):

(1) Source layer: the TEI document itself, XML-encoded with various depths of metadata derived from standard TEI modules.
(2) Display layer: a standard CSS/XHTML document derived from the source layer via XSL.
(3) Interactive layer: a JavaScript-driven, client-side application that allows a reader to interact with the display layer of the document to reveal aspects of content or metadata drawn from the original TEI encoding.

The TEIViewer begins simply with a modified branch of the existing TEI XSL stylesheet family. The addition of the interactive layer, as well as other modifications, is intended to maximize the capability for displaying metadata encoded in the source document that cannot readily be shown in human-readable form or that may cause clutter and interference for readers of the document's main text. The TEIViewer attempts to enable the publication of an individual TEI document as easily as possible by connecting together commonly available open source web tools, including the TEI XSL stylesheet family and the jQuery library, to form an application where extensibility is uncomplicated if it is desired, but is not required to produce a readable, interactive text from a standard P5-encoded XML document.

3 TEIViewer Functionality

The source layer of the TEIViewer is any TEI P5-conformant encoded document. To those who have worked with others' TEI documents or who read the TEI-L listserv, the last statement will seem to be naïve at best. And we acknowledge that the customized and interpretative aspects of TEI document encoding are two of the most significant impediments to the development of a simple, extensible TEI display tool.

Although the TEIViewer was originally conceived to display custom features for a particular TEI-based project, we determined that because of its relatively straightforward concept, we could make it generalizable to other TEI documents. One of our generalization strategies was to base the foundation of the display layer on the standard TEI XSL stylesheets. These stylesheets have evolved to provide elegant transformative display capabilities for most all of the elements in the standard TEI modules and therefore gave us a head start in addressing as broad a complement of documents as possible. Because we planned for the documents to be viewed over the Web using standard web browsers, and because we also planned interaction with elements of the documents via scripting, we chose for

the TEIViewer's display layer the most syntactically predictable output for the Web: XHTML.

When creating the display layer of the TEIViewer application, the source XML document is transformed using a customized version of the TEI stylesheet family's Common and XHTML branches. Excluding the import of an additional XSL stylesheet to create the code for the TEIViewer panel itself, the modifications made to the TEI stylesheets may generally be understood as falling into one of the following two categories:

(1) The modification of individual elements by moving display descriptions away from inline CSS styling and directives to CSS class name assignments instead.
(2) The inclusion of additional content, particularly structural content such as layers containing elements of the TEI metadata that may be enabled or disabled by the interactive TEIViewer panel.

Creating CSS classes for the representation of TEI elements instead of transforming them immediately via inline styles provides for two related outcomes that are essential to the function and extensibility of the TEIViewer. First, all appearances of similar TEI elements in the document may be recognized and treated as an independent class or unit by our interactive layer regardless of how similar they appear to a reader of the document. And second, the TEIViewer is able to change the appearance of those elements by modification of the Document Object Model (DOM) as defined by an external CSS file or JavaScript command, even after the document has been transformed and rendered, without 'mucking about' in the rules of the XSL stylesheets themselves. In other words, we attempt to abstract the specifics of display characteristics away from the source and transformation layer and into their own realm, controlled from a single, central location. By giving the display characteristics their own layer, we are providing an infrastructure of programmatically identifiable, and therefore more easily modifiable, elements of the document accessible through the standard DOM interface used by CSS and JavaScript applications.

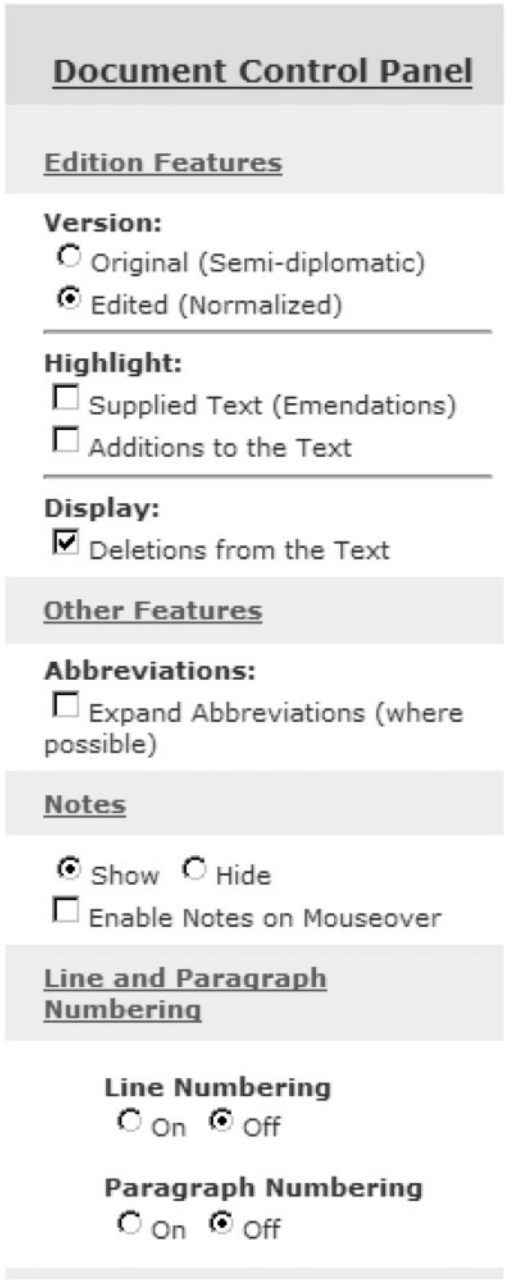

Fig. 1 Screenshot of expanded TEIViewer control panel

After the TEI-encoded source document is transformed by an XSL process into an XHTML document suitable for display by the web browser, its default appearance is rendered as described in

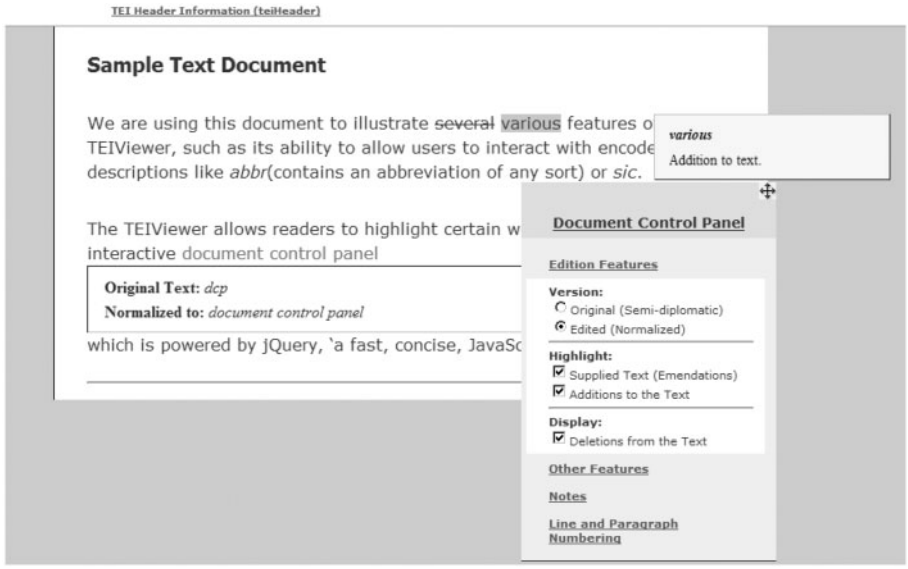

Fig. 2 Screenshot of the TEIViewer control panel with sample text

TEIViewer's custom CSS file. This CSS file[3] contains the display descriptions and parameters for all of the classes assigned in the customized XSL stylesheets (referred to above) in W3C standards-compliant, Levels 1 and 2 CSS. The display elements can all be modified in this single CSS file, with no programming knowledge necessary, creating custom display defaults for individual applications where desired (another benefit of converting inline styles in the XSL stylesheets to classes).

From a reader's perspective, the interactive layer is contained entirely within the collapsible, control panel-style menu (Figs 1 and 2), which uses standard web form widgets to allow the reader selectively to enable, disable, or highlight specific elements, features, or formatting within the document based on the CSS classes assigned from the source metadata during the XSL transformation. The control panel is driven by a powerful JavaScript library called jQuery, created by John Resig (2007).

Client-side interactions, such as those performed by the document control panel of the TEIViewer, can be a complicated process when dealing with many different browsers and platforms, and writing complicated routines to perform very common Web tasks is to reinvent the wheel forevermore. JavaScript libraries help to alleviate if not to eliminate these issues entirely by concentrating the resources of many developers on the fundamental challenges of browser and platform support and the development of standard methods and functions for common tasks. We chose jQuery because it handles traversal and modification of the DOM superbly, as well as having an open, chainable structure that makes using and writing plug-ins easy and efficient. It is also well-established and well-supported in the Web and open source communities at large, meaning many of the functions and plug-ins necessary for this application have already been written, tested, and made freely available for inclusion with the base jQuery library. The jQuery infrastructure enables us easily and straightforwardly to change the CSS display definitions of elements with the click of a button, live on the post-transformation document.

The live nature of the interaction is made possible via the Live Query plugin to jQuery, written by Brandon Aaron (2007). The Live Query plugin allows us to use HTML Form widgets to control

particular aspects of CSS classes by 'listening' for changes in the state of the Form controls. For instance, we can bind a particular function using Live Query to a particular checkbox widget in the document control panel of the TEIViewer that is written to control the highlighting of an 'addition' to the text, for example. Live Query signals the TEIViewer when a reader wishing to enable or disable the highlighting of text encoded as <add> checks the appropriate checkbox in the document control panel. Once that event is captured by the function we bound to the checkbox via Live Query, the control panel executes the TEIViewer's custom Javascript functions that control the CSS display characteristics of elements in the document that have a CSS class attribute value of 'addition' (assigned by the XSL transformation) via simple X-Path-style functions incorporated in the core of the jQuery library.

4 Implementation and Release

We expect the out-of-the-box application to be available for download from the project's website during fall 2008. Because the application displays an encoded document as a webpage and provides interactivity with the TEI elements described in Table 1, the tool can be used for Web display and interactivity as designed, but it can also be used simply to provide editors with a human readable version of their XML source during proofreading and review processes while encoding is still in-progress,[4] even in cases where a project may ultimately receive support for customized transformation and display. We hope it will therefore prove useful to encoders who wish to evaluate their work in real-time.

Because we envision changes and modifications as commonplace to project-specific implementations of the TEIViewer, we believe it will increase efficiency and simplicity for the tool to maintain a stylesheet that concentrates on the single, necessary output form: XHTML. Although the TEIViewer currently uses the TEI P5 XSL Stylesheets for the base of the display layer (as described above), we are working to develop a project specific stylesheet (still based largely on many features of the TEI stylesheets) that produces entirely XHTML display. We are also working to include additional TEI elements in the base implementation, focusing especially on elements that TEI users indicate would increase the utility of the interactive layer. The elements we include now are being grouped in a more modular fashion within the script, which will support users with customization, and we're planning to introduce a 'control panel builder', a web-based tool that will allow a user to choose which of the elements in the base implementation of the TEIViewer he or she would like to allow readers to interact with in the text.

Table 1 List of TEI elements managed by the control panel

Control panel designation	TEI elements controlled
Original	<abbr>, <add>, <corr> <del>, <expan>, <orig> <reg>, <sic>, <supplied>
Regularized/normalized	<add>, <corr>, <del> <orig>, <reg>, <sic>, <supplied>
Supplied	<supp>
Additions	<add>
Deletions	<del>
Abbreviations	<abbr>
Notes	<note>
Line numbering	<lb>
Paragraph numbering	<p>

5 Conclusions

While display is not a universal need, per se, it is the common, ultimate goal for many, if not all, projects. And while 'It may be the case that one general tool will never fit all possible uses for encoded documents' (PhiloLogic, 2008) or even respond to all of the customizations that the TEI supports, it is possible to envision a tool that facilitates Web delivery for many TEI projects, whether as a draft outcome, as a final outcome, or as an in-progress outcome that aids in the review process during the encoding stage. We believe that by leveraging popular, community-driven open source libraries, it will be possible for a few individuals to continue

maintaining and enhancing the TEIViewer. Our aim is to continue to develop the TEIViewer toward this end.

Finally, at the 2007 TEI Members Meeting, Schreibman *et al*. raised an important question, one that merits restatement here: 'Is digital humanities tool development worth the effort?' (2007). Our answer is a resounding, qualified *yes*. Developing the TEIViewer has presented us with the opportunity to consider more carefully the audience and purpose of our own work and to explore the theoretical and practical implications of how and why we share it with others. Further, it has given us pause to think about one aspect of the mission of the TEI: 'to develop and maintain a set of high-quality guidelines [. . .], and to support their use by a wide community of projects, institutions, and individuals' (2007b). Tool development addresses the latter aspect of this mission statement. If they are well-designed, meet needs, and easily accessible, tools can support and encourage the use of the TEI Guidelines. While others are taking important steps to answer questions about tool development from the perspective of tool developers,[5] perhaps as we look ahead, we should more carefully take into consideration the user perspective, asking TEI-users and potential users how tool developers can address their needs and, in doing so, facilitate the use of the Guidelines.

References

Aaron, B. (2007). *jQuery Live Query Plugin 1.0*. http://brandonaaron.net/code (accessed 26 September 2007).

Cohen, M. (2004). *Separation: The Web Designer's Dilemma. A List Apart*. http://www.alistapart.com/articles/separationdilemma (accessed 10 March 2009).

Hagel, S. (1997). *Classical Text Editor*. http://www.oeaw.ac.at/kvk/cte/ (accessed 10 March 2009).

PhiloLogic. (2008). *What is PhiloLogic?* http://philologic.uchicago.edu/ (accessed 21 January 2008).

Resig, J. (2007). *jQuery*. http://jquery.com (accessed 25 October 2007).

Schreibman, S., Hanlon, A., Daugherty, S., and Ross, A. (2007). The Versioning Machine 3.1: Developing Tools for TEI. *TEI@20: 20 Years of Supporting the Digital Humanities*. College Park, Maryland, 31 October–3 November 2007.

Schreibman, S., Kumar, A., and McDonald, J. (2003). The versioning machine. *Literary and Linguistic Computing*, **18**(1): 101–7.

<teiPublisher>. (2004). http://teipublisher.sourceforge.net/docs/ (accessed 10 March 2009).

TEI: The Text Encoding Initiative. (2007a). *Minutes of the TEI Presentation Tools SIG Meeting in Nancy, 08 November 2003*. http://www.tei-c.org/Activities/SIG/Tools/pt01.xml (accessed 10 March 2009).

TEI: The Text Encoding Initiative. (2007b). *TEI: Goals and Mission*. http://www.tei-c.org/About/mission.xml (accessed 10 March 2009).

TEI: The Text Encoding Initiative. (2007c). *TEI Presentation Tools SIG Minutes: 3 November 2007*. http://www.tei-c.org/Activities/SIG/Tools/pt04.xml (accessed 10 March 2009).

TEI: The Text Encoding Initiative. (2007d). *TEI: Tools*. http://www.tei-c.org/Tools/ (accessed 10 March 2009).

TEIWiki. (2008). *TEI Tools Wiki*. http://www.teic.org/wiki/index.php/Category:Tools (accessed 19 January 2008).

Tufte, E. (2001). *The Visual Display of Quantitative Information*. Cheshire: Graphics Press.

UW Technology. (1999). *Graphics and Web Design Based on Edward Tufte's Principles*. http://www.washington.edu/computing/training/560/zz-tufte.html (accessed 10 March 2009).

Notes

1 In July 2008, Threepress Search, an 'e-pub tool', was added to the Tools Wiki page. Information about this resource is available at: http://code.google.com/p/epub-tools/. With the exception of MarkLogic Server, all of the tools discussed are open source and freely available to users.

2 Although we have distinguished only two broad categories in this discussion, additional categories can and should be delineated. For instance, the Image Markup Tool and Sacodeyl Annotator are both designed to be accessible to users of all skill levels, from beginner to expert. Moreover, both fit into the more distinct category of 'markup' or 'editing' tools, which is currently listed as a sub-group of the larger tools category. XML database tools such as eXist and MarkLogic Server, however, are much less accessible to beginners, and

both are listed within the 'Querying tools' subcategory of the larger tools grouping as well.

3 The CSS file also serves as a central point of documentation for the display characteristics of individual elements, which had previously been available only by viewing the source code of the XSL stylesheets used to create the document.

4 During the tool's design and development stages, we tested it with a TEI P5-conformant edition of *Hafgeirs saga Flateyings*, an 18th-century Icelandic saga manuscript. We found viewing and interacting with transformations of the edition's XML source prior to the completion of the encoding stage invaluable, especially where complex encoding and interpretive decisions such as <choice> tag sequences and <note> tag content were involved.

5 We refer readers to Susan Schreibman and Ann Hanlon's *Digital Humanities Tools Developers' Survey*. A discussion of the findings of the survey, *Determining Value for Digital Humanities Tools*, was presented at the 2008 Digital Humanities Conference in Oulu, Finland.