Journal of the Text Encoding Initiative

Issue 2 | February 2012
Selected Papers from the 2010 TEI Conference

TEICHI and the Tools Paradox

Developing a Publishing Framework for Digital Editions

Sebastian Pape, Christof Schöch and Lutz Wegner

Electronic version
URL: http://journals.openedition.org/jtei/432
DOI: 10.4000/jtei.432
ISSN: 2162-5603

Publisher
TEI Consortium

Electronic reference
Sebastian Pape, Christof Schöch and Lutz Wegner, « TEICHI and the Tools Paradox », *Journal of the Text Encoding Initiative* [Online], Issue 2 | February 2012, Online since 03 February 2012, connection on 22 April 2019. URL : http://journals.openedition.org/jtei/432 ; DOI : 10.4000/jtei.432

This text was automatically generated on 22 April 2019.

TEI Consortium 2012 (Creative Commons Attribution-NoDerivs 3.0 Unported License)

TEICHI and the Tools Paradox

Developing a Publishing Framework for Digital Editions

Sebastian Pape, Christof Schöch and Lutz Wegner

AUTHOR'S NOTE

We gladly acknowledge the work of Dmitrij Funkner, Mohammed Abuhaish, and Roman Kominek who took on the implementation and documentation of this interdisciplinary project. We also acknowledge previous work done by Weam Abu Zaki on the XML Content module for Drupal on which our work has built. We would like to thank Susan Schreibman and Alexander Czmiel for sharing insight into their respective tools with us. Finally, we would like to acknowledge the expert advice of the anonymous reviewers who helped us clarify several aspects of this paper.

1. Introduction

1 In their recent overview of publishing and delivery tools for TEI files, Stephanie Schlitz and Garrik Bodine emphasize the need for tools which allow textual scholars, especially those new to Digital Humanities and TEI, to make their text editions available online (Schlitz and Bodine 2009, sec. 2). Very recently, Marjorie Burghart and Malte Rehbein have conducted a survey among users of TEI specifically interested in manuscripts. They conclude that one of the biggest obstacles for scholars beginning work with TEI was a lack of "user-friendly, adapted tools facilitating the processing, analysis and publishing of the material", resulting in a perceived gap between encoding documents and publishing them online (Burghart and Rehbein 2011). It seems that more and more textual scholars are aware of the powerful encoding mechanisms that the TEI provides and wish to use them for their work; however, many of them do not have the technical expertise, support, and/or infrastructure needed to employ one of the existing advanced repository and delivery tools, such as the ones based on the Fedora Commons Repository Software or eXist-db.[1]

2 Fortunately, not all projects require that kind of advanced infrastructure provided by the Fedora Commons Repository Software or eXist-db, and a number of tools have been developed specifically with ease of publishing TEI documents in mind. The most mature of these tools is without doubt the Versioning Machine, "a framework and an interface for displaying multiple versions of text" first released in 2002 and now available in version 4.0, developed by a number of people led by Susan Schreibman.[2] A more recent tool is the Scalable Architecture for Digital Editions (SADE), which "tries to meet the requirement for an easy to use publication system for electronic resources and Digital Editions" and was developed by a team around Alexander Czmiel.[3] Another fairly new tool is the TEI Viewer, built for display and interactivity and developed by Stephanie Schlitz and Garrik Bodine.[4] The most recent tool, released in August 2010, is TEI Display, a plugin developed by the Scholars' Lab at the University of Virginia Library for use in the digital archives and exhibits system Omeka.[5] Unfortunately, for some of the existing tools there seems to be no continuous support and maintenance. This is presumably the case with TEI Viewer, with the quite powerful but complex <teiPublisher> (working in conjunction with eXist and Lucene),[6] and with the somewhat simpler XML Content module for Drupal, which focuses on the online validation and display of XML documents.[7]

3 In this paper, we present our own publishing framework, called TEICHI ("Text Encoding Initiative Computer-Human Interaction"). TEICHI began development in 2009 as an interdisciplinary project involving researchers and graduate students from both literature and computer science. At the core of the framework is Drupal (version 7), a popular open-source, fairly powerful, and modular content management system (CMS) that serves as a repository for the TEI documents. Because the development of Drupal has been done with "content creators" and "non-technical site builders" in mind, it may be particularly well-suited for our target users.[8] Indeed, most humanities scholars we are targeting with TEICHI probably fall into either one of these two categories since they want to publish individual scholarly editions with a small team and keep close control of the website for their scholarly edition. We therefore created a set of Drupal modules with an eye towards making them as easy to use as possible. Currently, these modules include *TEI Content*, *TEI Download* and *TEI Search*.[9]

4 Of course, the perceived ease of use depends on the role that a person plays in a project. We have identified four user roles for any given tool for publishing scholarly editions: (1) the *user* of the published edition who encounters the interface and features of the edition and is focused on usability and aesthetics; (2) the scholarly *editor* primarily concerned with transcribing, annotating, uploading and updating documents; (3) the *administrator* whose task it is to set up the server and possibly customize the tool for the editor; and (4) the *programmer* who may wish to more deeply customize, enhance, or expand the tool or to make it compatible with new technologies. In the absence of research on the requirements of humanities scholars fulfilling these roles, we will define those requirements which seem most relevant to us.[10] A user may place himself or herself in more than one of these roles. In fact, this will almost always be the case precisely for those scholars with smaller edition projects that we are targeting with this tool: they will not only take up the role of the editor, but may also want to execute administrator tasks without support from an expert. Many of them will no doubt need to make minor changes or adjustments to adapt the tool to their specific project needs, acting as a programmer. When choosing a publishing tool for an online edition, user features should be considered

along with reader features, which is why any discussion of publishing tools should, in our view, include an analysis of these four perspectives.

5 To showcase the particular strengths and possible use cases of our framework, and to substantiate claims regarding ease of use of our tool set, we compare TEICHI with the VM and with SADE. Each of them takes quite a different approach to the tasks at hand, but both are open-source, freely available, and currently maintained publishing frameworks, with a focus on being easy to use and requiring comparatively little technical support.[11] (See table 1 for an overview over some of the basic features of the three tools.) Clearly, our comparison between TEICHI, the VM, and SADE is somewhat subjective. It is presented here mainly for the purpose of showing where TEICHI is positioned in the world of tools for the online display of TEI-encoded documents, and for which specific use cases each of the three tools is particularly well suited.[12]

	TEICHI 0.6	**VM 4.0**	**SADE**
Target projects	Digital editions of printed material with a simple editorial situation	Digital editions of shorter material, such as poetry, with many textual witnesses	Flexible, depending on setup; also for bilingual editions
Basic layout	One text column, tools and notes in the margins, toolbar at the bottom	Any number of columns displaying textual witnesses or bibliographic metadata	Flexible: one or two text columns plus navigation and index/ search sidebars
Relation to website infrastructure	Modules integrated into Drupal 7	Can be integrated into any website or used offline	Provides full website infrastructure
Transformation	Takes place on the server	Takes place in the browser or is done prior to publication	Takes place on the server
Support for TEI elements	Majority of TEI Lite, plus specific support for the 'choice' element	Parallel segmentation and location-referenced encoding, most editorial interventions, and various types of notes	Flexible, depending on custom XSLT and CSS files

Table 1: Comparison of TEICHI, the VM, and SADE

6 Following our analysis and comparison, we take a step back and review the situation for frameworks aimed at digital editions in general. This situation seems to be marked by what could be called the "tools paradox": the need for such tools is quite urgent, and there are several tools available; however, the number of actual implementations of these tools by textual scholars is still relatively low. Are the available tools too cumbersome to use or are they unsuited for digital scholarly editions, and what can be done about it? Is it enough to develop tools which do the job they were intended to do, or is it equally important to build up a strong user base?[13] Is there a vicious circle, a self-fulfilling prophecy? After all, a small user base doesn't warrant maintenance and further

development, causing scholars to be reluctant to join the bandwagon because the project seems stale and lacks support (Schreibman et al. 2007). If this is the case, what can then be done to make frameworks both easier to use for a variety of users and yet simple enough to maintain and remain open ended in their development?

2. Reading and Viewing: The User's Perspective

7 We would like to start with the user perspective, in which several elements come into play: the overall look and feel of the site, the navigational aids, the display features of the text, the interaction that is possible, and the information retrieval tools provided.

8 Because TEICHI is fully integrated into Drupal, the overall layout depends on the theme chosen and the block regions used. TEICHI has proven to be compatible with the Garland and Bartik themes, each offering fully customizable color schemes and providing numerous block regions. Apart from the header and footer blocks, TEICHI uses a one-column layout for the main text with a large margin to the right where the notes are displayed. Additional tools for download and search may be placed in any of the block regions. There is a fixed bottom toolbar providing some of the navigational aids and interactive features. This basic layout is shown here on the basis of an implementation of TEICHI we undertook (see fig. 1 and Bérardier de Bataut 2010).

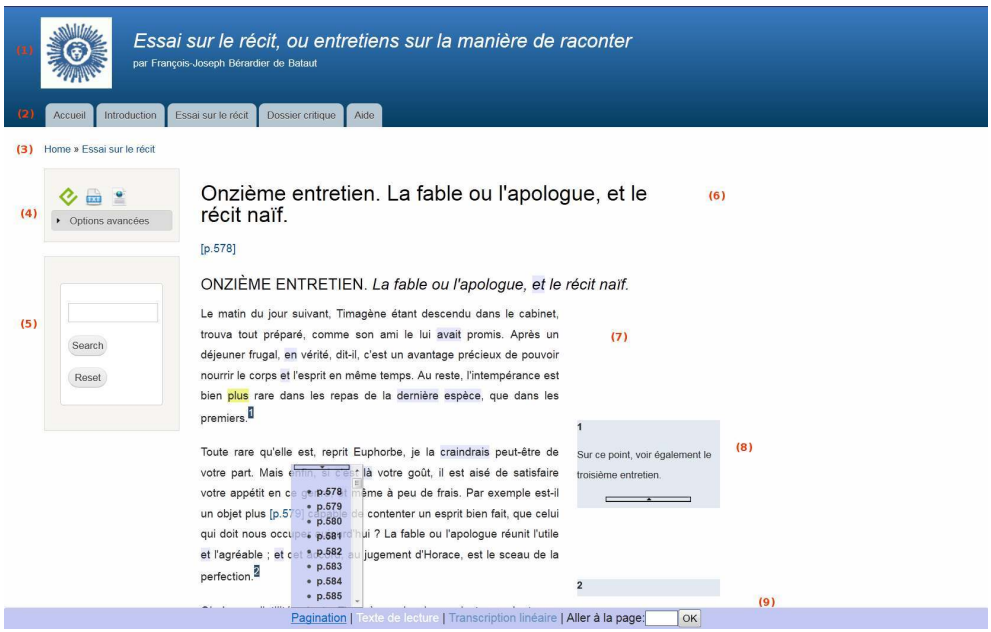

Figure 1: Page layout and features of TEICHI. (1) Header region of the website (customizable). (2) Tabbed navigation (customizable). (3) Breadcrumb navigation (optional). (4) Download section. (5) Direct search section. (6) Chapter or section heading. (7) Main text column with page numbers and highlighted variant words. (8) Editor's note, here shown unfolded. (9) Function toolbar: in-chapter page navigation (left, unfolded), choice of two text renderings (middle), and "go to page" feature (right).

9 Text in TEICHI is structured not into individual pages, but into chapters or sections ordered hierarchically as in the TEI text division structure. Navigation inside these books can be from one chapter or section to the next or previous, as well as one level up. The TEICHI toolbar provides additional navigational aids: a page index of the current chapter or section and a "go to" feature allowing users to go to any page of the current book. Page numbers can be displayed (inline), but there is no option for line numbering. Digital facsimiles of individual pages can be viewed by clicking on the page numbers, but this

feature is very basic; there is no two-column layout or pop-up image viewer that conveniently permits comparison of the facsimile and transcription. Text display is focused on one main column with author and/or editor notes placed in the right margin and collapsed by default to avoid an overly cluttered page. The user can toggle between two views of the text: one linear transcription of the text and one modernized, corrected, or otherwise regularized reading text. All words affected by variation between these two transcriptions are highlighted by colored backgrounds. Apart from the opening or closing of notes and the navigational aids, interaction with the edition in TEICHI primarily concerns this view toggle option.

10 Users can retrieve the texts from the online edition via a download block that provides various download options: an EPUB version for offline reading with an e-book reader or smartphone, a plain text version for use with basic text analysis tools, and the TEI source for further use with advanced XML-aware text analysis tools. For all of these file formats, the user can also choose between various renderings of the text: either the linear transcription or the reading text, with or without editorial notes, or a single chapter or several chapters, placed either in separate files or in one single file.

11 When comparing TEICHI to the VM, the first thing any user will probably notice is that the VM allows any number of columns of text next to each other, while TEICHI provides only one. Second, interaction with the edition is much greater in the VM, since the user can choose what to display in any of the columns, be it metadata, critical notes, or any of available versions of a text. Also, the user of the VM can decide whether to display line numbers and whether to display notes inline or as mouse-over popup notes. Individual lines can also be highlighted in all of the currently displayed versions of a text, helping to compare them. There is an image viewer and files can be downloaded directly from the browser. The VM does not offer any support for building a website around the edited texts; instead, it must be integrated into an external website. All of the existing implementations of the VM are quite similar in layout and with regard to the features offered (see, for instance, Freytag-Loringhoven [2009] in fig. 2, or the Digital Ælfric

Project [2011], the latter providing no less than 17 witnesses of some of Ælfric's *Catholic Homilies*).

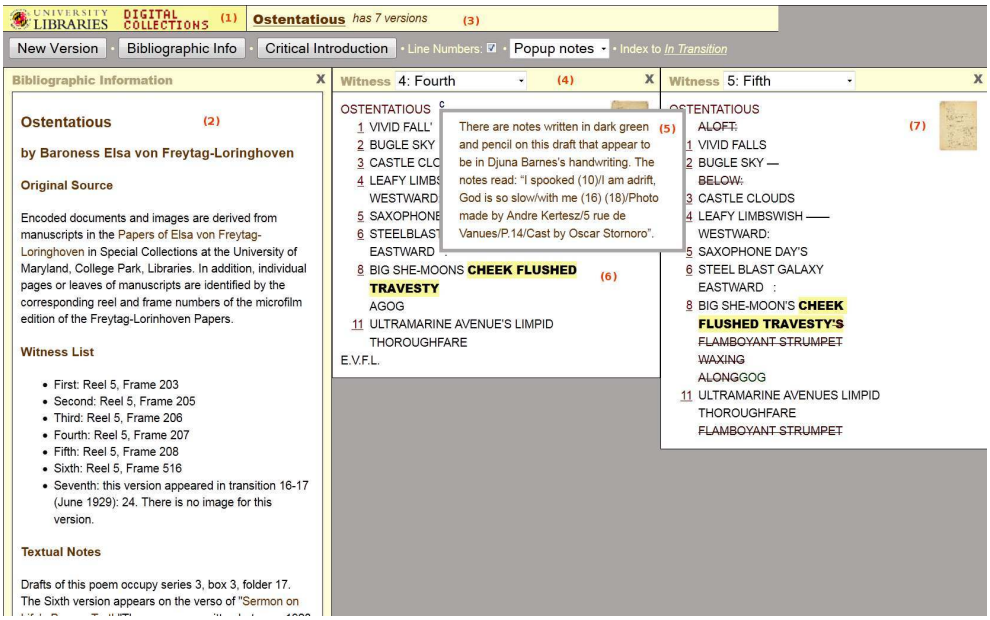

Figure 2: Page layout and features of the VM. (1) Header region (left) and interactive toolbar for managing the display options (below). (2) Column for bibliographic data, from `<teiHeader>`. (3) Title of work displayed, with number of versions. (4) Selection tool for available witnesses. (5) Popup note, shown here expanded. (6) Main text column, shown here with line numbering, highlighted line, and scribal corrections. (7) Thumbnail of facsimile, which, when clicked, opens the image viewer.

12 When comparing TEICHI to SADE one is struck by the fact that almost every implementation of SADE has a distinct layout, structure, and feature set. For instance, the correspondence of Alexander von Humboldt and Christian Gottfried Ehrenberg (Humboldt and Ehrenberg 2008) features a four-region layout (see fig. 3). Editorial interventions and notes are displayed in a rather static and conventional way. Its outstanding feature is the advanced index, a register of names of people, places, and zoological terms. Each indexed term is highlighted in the text and can be used to retrieve the relevant index entry in the right sidebar, which links to all instances of this term. This allows for a non-linear reading of the text corpus that is much faster and more convenient than that offered by an index in a print edition. A contrasting example is the *Inscriptiones Graecae* edition which provides a two-column layout for displaying the Greek and German versions of the *Inscriptiones*, with a convenient synchronous page flip mechanism and an advanced search allowing a variety of focused searches (Inscriptiones Graecae 2011). From a user perspective, these two implementations of SADE actually have very little in common.[14]

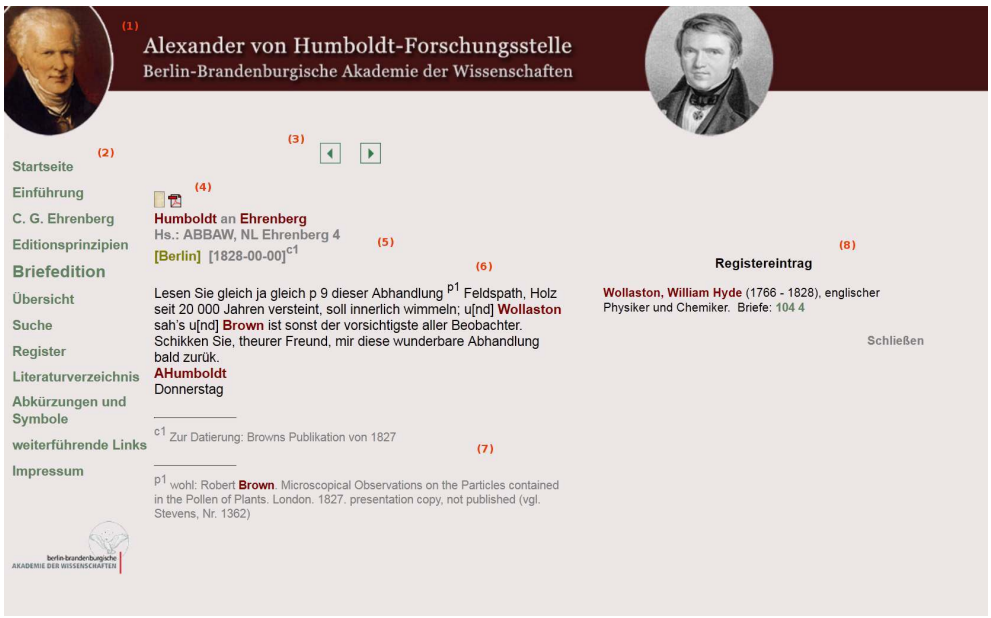

Figure 3: Page layout and features of SADE. (1) Header region of the edition or archive website (customizable). (2) Navigation sidebar. (3) Page navigation. (4) Facsimile view and PDF download section. (5) Basic metadata of the current letter. (6) Main text column. (7) Footnote area. (8) Index function, with links to relevant letters.

13 It appears from this comparison that SADE is highly customizable and hence suitable for a variety of use cases, while TEICHI and the VM are each best suited to the display of specific types of documents. TEICHI is designed for editions dealing with multiple relatively long non-manuscript texts, each with a straightforward editorial situation, where the editor chooses to present only one print edition. The VM favors shorter print or manuscript texts, especially poems of which many differing versions exist and which can be displayed and compared with great convenience. From a user perspective SADE's ability to accommodate a wide range of editions does not overwhelm the user for any particular document or intended use, while an advantage of the VM and, arguably, of TEICHI is that they provide particularly neat solutions for specific use cases.

3. Encoding: The Editor's Perspective

14 From the editor's perspective, the most important features of a framework for digital editions are the supported encoding schemes or tagsets, the ease of the publishing and updating process, a short turn-around between changes made and changes seen, the extent of documentation or support available, and the number and variety of sample installations.[15]

15 TEICHI's support of TEI encoding schemes is currently limited to a subset of TEI Lite, although this situation is expected to change in the near future. The principal aim of TEICHI is to provide the editor with a means of encoding and displaying a linear transcription of the source text. Typically this type of edition retains many particularities of the source while also documenting and/or displaying a certain degree of correction, regularization, and modernization. This is primarily achieved through the use of the well-known and simple `<choice>` mechanism of TEI. This means that although only one edition of a given text can be represented, two alternative readings of it can be displayed. Also, TEICHI allows the editor to add authorial and editorial annotations in an

unobtrusive way by means of the <note> tag. Moreover, TEICHI extracts information on page numbering from the TEI documents and uses this automatically to provide help navigating longer texts.

16 TEICHI provides full integration into the Drupal administration GUI: this means that uploading new documents or updating and organizing existing ones (and even fonts) is done through this GUI. Also, setting some of the display options, such as the colors for highlighting of variants or the text displayed on a number of buttons, can be done without delving into any code (see fig. 4). TEICHI also provides documentation to editors and administrators on how to install, use, and customize the modules. Moreover, the documentation helps with the encoding itself as well as with the customization of the stylesheet should the editor wish to tweak the default display or add elements to it.[16]

Figure 4: Setting color options

17 In contrast to the rather basic support for editorial interventions and textual variants of TEICHI (which does not go beyond the possibilities offered by TEI Lite), the VM supports TEI's very powerful Text Criticism module, which is essentially based either on the parallel segmentation or the internal location-referenced method of encoding described in the TEI Guidelines (Burnard and Bauman 2007, sec. 19.2.3). This is an efficient and precise, but relatively complex, way of encoding multiple versions of a text. From an editorial point of view, it has the benefit of not requiring the encoder to declare a base text (Schreibman et al. 2007). While the VM does not make the locus and extent of variation between versions explicit, the tool's advanced display options easily make up for this. The VM provides quite extensive documentation, especially on encoding documents but also concerning internal file and folder structure and customization options of the tool. In addition, there is a form for bug reports on the project website.

18 Compared to TEICHI and the VM, SADE again offers a much more open feature set. Basically, any TEI-compliant XML document can be used because SADE comes with support for the TEI's own stylesheet library, thus accommodating a much more diverse scope of encoding needs and possible use cases. However, this comes at the expense of lacking the "out-of-the-box support" that TEICHI and the VM provide for those features and TEI modules that fall into their specific application domains.

19 Thus, from the editor's perspective, TEICHI and the VM cater directly to their specific needs and have a clear scope in terms of tag coverage. While SADE does not have any built-in limitations, it requires customization for any given project and comes with a steeper learning curve. Before we move on to the administrator's perspective, we summarize our comparison in table 2.

	TEICHI 0.6	VM 4.0	SADE
Page layout	1 text column, plus 1–2 sidebars, depending on setup	Any number of columns, adjustable by user	Flexible: 1–2 text columns, 0–2 sidebars, depending on setup
Text structure	One page per chapter or section; page numbers	One page per document; line numbers	Flexible: page-by-page or one page per section or text
Multiple text versions	Toggle mechanism via *<choice>*	Any number of transcriptions via parallel segmentation or internal location-referencing	One or two transcriptions side-by-side
Annotation	Author's and editor's notes, collapsible, in sidebar	Four types of notes, as tooltip or in separate column	In right sidebar by default; depends on setup
Digital facsimiles	Very basic	Popup image viewer	Inline image viewer, depending on setup
Search	TEI-aware	No	TEI-aware
Download formats	EPUB, TXT, XML/TEI via download block	XML, HTML via browser	HTML, PDF, EPUB, DOC via icons

Table 2

4. Installation: The Administrator's Perspective

20 From the administrator's perspective, the most important aspects are the hardware and software requirements, the ease of setting up the server, installing the tools, customizing them, and having support for all these tasks including some type of documentation.

21 As TEICHI is a set of modules for Drupal, one of the many content management systems written in PHP, a webserver (e.g. Apache or the IIS) with PHP 5.2.5 or higher and MySQL 5.0.15 or higher is required. Most shared webhosting services provide this. Setting up Drupal itself is not entirely trivial, but the procedure is well-documented on Drupal's website.[17] Installing the TEICHI modules is then fairly easy. It can be done entirely via Drupal's administration GUI and following normal and well-documented module installation procedures. All Drupal settings required for TEICHI are documented and are facilitated by Drupal's GUI. After installing the module(s), the administrator is able to modify the basic layout, colors, and font sizes, as well as menu and error texts, via the module's generic Drupal administration menu. Though there are limits to the adaptability, manual editing of a CSS file is usually unnecessary.

22 The VM has even lower system requirements than TEICHI since it does not require a webserver and thus can be used offline. Installing the VM is straightforward since there is basically no installation procedure other than unzipping the files and copying the XML/TEI or HTML documents into the right folder. Following that, the files can be opened in a browser. On the other hand, making the documents publicly available requires a webserver. Besides the webserver itself, setting up the VM for online use should not pose any major challenges. The drawback is that the organization and structure of the website are left to the administrator. Another downside is that either the TEI files must be transformed in advance to HTML, or the user's browser must perform the XSLT conversion. No help can be expected from the VM running on a webserver for this transformation.

23 SADE requires one of the latest versions of the Java SE Development Kit for the application server Jetty, which it includes. SADE is distributed as a ZIP archive containing ready-to-use install files, including the Jetty server and eXist-db components. This means that SADE can easily be used on a dedicated server but is not as easily deployed in shared hosting environments. Also, setting up SADE is somewhat challenging and the average humanities scholar will probably not be able to do this without support, since several fairly complex technologies must be made to work together. Another difficulty arises from the fact that documentation concerning SADE is very limited. Like TEICHI (through Drupal) and unlike the VM, SADE offers many options to build comprehensive websites.

24 From the administrator's perspective, then, it appears that the VM is the tool with the lowest demands, followed closely by TEICHI. SADE is clearly the most complex and challenging publishing framework. Note also that despite the divergent server-side requirements, all projects require JavaScript-capable browsers on the client side.

5. Tool Building: The Programmer's Perspective

25 From the programmer's perspective, it is important to look at the tool's architecture. Each tool has a particular way of implementing its features, and its architecture determines how easy it is to integrate digital editions into a broader website environment. The first part of this section gives a rough overview of how TEICHI works and is mainly meant for readers not familiar with Drupal. Due to the specifications of the Drupal framework and the use of the standard method of transforming TEI to HTML by employing an XSLT processor, this part of the section will hardly contain any surprises for advanced readers. The second part of the section then shortly compares the differences in the tools' chosen implementations.

26 In order to understand how TEICHI is implemented, a short introduction to some basic Drupal concepts is needed. The relevant terms are *nodes*, *hooks*, *filters*, and *blocks*. Basically, Drupal stores all content as *nodes* irrespective of the type of content. Different content types then allow changing the way each node is displayed. For instance, blog entries will have a different layout from book pages. *Hooking* is a well-known software engineering technique which in this case allows modules to interact with Drupal's core. Hooks in Drupal are simply PHP functions named according to a predefined scheme. At certain points, Drupal allows intervention from modules, calling these hooks in all enabled modules that implement it. For example, when a module is installed or deleted, the corresponding hooks are called.[18]

27 Drupal stores input in a raw format in its database. Then, prior to displaying any output in a browser, it processes the content by applying rules defined for each input format. Input formats themselves are represented by an ordered collection of filters where each filter feeds its output to the next filter. Filters are simply sets of rules which are applied to transform their input in some way. If used wisely, this concept allows great flexibility by defining simple filters and making them work together. Within this paradigm, TEICHI is such a filter implementing the transformation from TEI/XML to XHTML by an XSLT processor. By assigning CSS classes to certain HTML elements corresponding to their original XML elements, it is then possible to determine the desired layout. Further functionality, like switching between variant readings and navigating to original page numbers, is implemented by adding JavaScript. The largest advantage of this approach is that all information within the original data is kept. This is especially useful for TEI documents since different transformations to various output formats are thus possible.

28 Besides scripts run for installing and uninstalling, the TEICHI main module contains JavaScript files, the XSL stylesheet for rendering, and some files needed for Drupal. The main file with the Drupal-compliant name *teichi.module* contains PHP functions. Many of those functions use the previously described hooks. The most important of them is *teichi_filter_info* (see fig. 5), which determines which function is called when a node with content type *teichi* is shown.

```
function teichi_filter_info() {
 $filters[‘teichi’] = array(
   ‘title’ => t(‘XSL filter for TEIChi module’),
   ‘process callback’ => ‘_teichi_transform’,
   ‘tips callback’ => ‘_teichi_filter_tips’,
   ‘cache’ => FALSE,
 );
 return $filters;
}
```

Figure 5

29 The function returns an array which refers to *teichi_transform*. Consequently, this function calls the XSLT processor. Although the hook described above had to be adjusted for Drupal 7, the transformation function itself relies on the *XML Content* module transformation function. However, one of the differences is that, due to the already determined input format (TEI Lite), the XSLT file is pre-determined and thus *_teichi_transform* only needs to receive the XML to transform it. This way the user does not need to worry about the configuration of the XSLT processor nor select a stylesheet beforehand. Furthermore, CSS and JavaScript files are added to the output via the Drupal API with *drupal_add_css()* and *drupal_add_js()*, respectively. Figure 6 shows the function that calls the XSLT processor.[19]

```
function _teichi_transform($xml, ...){
  // some error checks
  ...

  // set TEI XSL file,
  $path_to_xslt = drupal_get_path('module','teichi').'/xsl/drp_style.xsl';

  // add Javascript files
  ...
  drupal_add_js(drupal_get_path('module','teichi').'/js/drp_main.js');

  // add CSS file
  ...
  drupal_add_css($cssPath, $options);

  // Transformation
  // Load the documents
  $dom = new DomDocument('1.0', 'UTF-8');
  $dom->loadXML($xml); ...
  $xsl = new DomDocument('1.0', 'UTF-8');
  $xsl->load($path_to_xslt);

  // Create the XSLT processor and transform
  $proc = new XsltProcessor();
  $xsl = $proc->importStylesheet($xsl);
  $newdom = $proc->transformToDoc($dom);

  $out = $newdom->saveXML();
  return $out;
}
```

Figure 6

30 There are also numerous functions providing uploads for TEI/XML files or fonts and for making adjustments to TEICHI's configuration. It is worthwhile mentioning that the *TEI Search* module adds a second filter which gets the output of the main module's filter and changes it, e.g. to highlight searched words.

31 Besides the main content handled via filters, modules may also provide additional content or features. These are made available as blocks which can be positioned inside the theme(s) used. The *TEI Download* module provides a block with a configurable set of items. These let the user download the node content in various formats: EPUB, plain text, or TEI.

32 To implement the block, two functions are needed which make use of Drupal's hooks: *teichidownload_block_info()*, with the block's basic definition, and *teichidownload_block_view ()*, which defines the block's content. The latter first tests whether the actual node type is a *teichi* node and whether the user has sufficient rights to view the block. If this

requirement is fulfilled, a generic Drupal function is called which creates the block's content by invoking the module's form constructor function. If the user picks one of the download icons and submits the form, the form's values are validated and then passed to the *teichidownload_form_submit()* function (see fig. 7). Depending on the value submitted, the appropriate function is called to generate the document to download.

```
function teidownload_form_submit($form, &$form_state) {
 $nid = arg(1);
 $button = $form_state['clicked_button']['#name'];
 switch ($button) {
  case 'epub_download_button':
    _teidownload_chapter_as_epub($nid);
    break;
    ...
 }
}
```

Figure 7

33 As an example, we briefly describe the conversion to the EPUB format. Since EPUB is basically a special ZIP file, the function opens a ZIP archive and first writes some metadata to it. The content of an EPUB file is XHTML formatted by a CSS file. Since e-book readers currently do not support JavaScript, no interaction besides turning pages is possible. The user has to decide which version of the available texts he or she wants to view (see above and fig. 8). Then, the appropriate CSS file is selected and written to the ZIP archive. Finally, the body of the text, a cover image for the title page, and a table of contents for the EPUB are added. Before the file is ready for download, the node's TEI data is again transformed with the XSLT processor and written to the archive. While the module supports conversion to EPUB and plain text (and allows downloading the original TEI file), this conversion mechanism can of course be used to cover various formats.

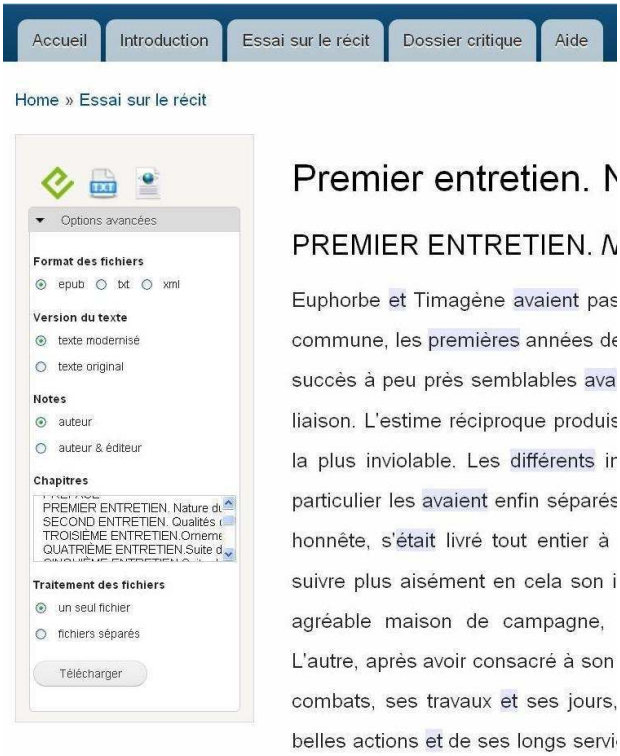

Figure 8: The TEI Download module

34 Comparing TEICHI with the VM and SADE, we first note that all tools reviewed here transform XML/TEI-documents to HTML or XHTML for visualization and make use of JavaScript for further functionality. However, the projects differ in how they implement this concept. TEICHI makes use of the libxslt library for PHP, which in turn employs the XSLT C library for GNOME as XSLT processor.[20] The VM provides only an XLST file as well as all necessary CSS and JavaScript files but leaves it to the editor or user to transform the document in his editor or browser (respectively). SADE is realized as a web application using Saxon (which is also written in Java) to transform the documents.[21] Another difference concerns the storage of the TEI files to be processed. TEICHI stores them in the database provided by Drupal and requires SQL to retrieve them. The VM simply stores them in the underlying filesystem, and SADE uses a native XML database (eXist-db) and therefore employs XQuery.[22]

35 We conclude the comparison of the projects in table 3, which gives an overview of the features discussed here.

	TEICHI 0.6	VM 4.0	SADE
License	GPL	GPL	[free][23]
Requirements	Webserver, PHP 5.2.5+, MySQL 5.0.15+	Webserver	JDK 5+
Language	PHP	XLST only	Java
Database	SQL	Filesystem	eXist
XSLT processor	XSLT C library for Gnome (supports XSLT 1.0)[24]	[none]	Saxon XSLT Processor (supports XSLT 2.0)

Realization	Drupal module	standalone	Jetty web application

Table 3

6. Conclusion and Future Work

36 Our overview and comparison of tools yields a twofold result. First, it appears that tools like TEICHI or the VM provide feature sets suitable for quite specific use cases and cater to users with quite specific needs and competencies. Tools like SADE are much more flexible and adaptive and can be used for various types of projects. Second, TEICHI and the VM are quite easy to install, use, and maintain, while SADE demands considerable preliminary customization and more IT skills of administrators and/or programmers.

37 At first glance, the source of the "tools paradox" mentioned earlier would seem to be that there are essentially two competing and mutually exclusive ways of broadening the user base of any publishing tool: making the tool as generic, flexible, and customizable as possible versus making it as easy as possible to use out-of-the-box. The conclusion would be that either "complexity is the price paid for generality" or that "limitation is the price paid for ease-of-use," with both standing in the way of a widespread user base. However, the challenge to combine expressive power with simplicity of use is faced by any developer in the domains of application software or operating systems, and it marks the long-term evolution of such endeavors. The solution to this tension is a layered software architecture with a user-friendly outer shell that follows a common graphical user interface convention. In this way, all those users interested in a simple out-of-the-box solution will be provided with default features and a default configuration without compromising the overall flexibility of the software. There is no reason why frameworks for digital editions shouldn't follow the same evolutionary path as other end-user application software. TEI is a good starting point since it separates content from appearance, the latter delegated to stylesheets. A CMS can serve as an interface between databases and programs and provide a basis on which to build further layers for customization. A modular approach allows user communities with specific needs to develop additional modules with customized feature sets without burdening editors and administrators with unnecessary complexity for features they do not need.

38 As mentioned earlier, we do not intend our analytic separation of the roles of user, editor, administrator, and programmer to exclude any one individual taking on more than one of these roles. So-called "end-user software development" is on the increase, in which people with domain-specific knowledge and individual requirements take on tasks originally delegated to software professionals. As a consequence they are faced with the problems software developers routinely deal with, such as design, reuse, integration, testing, and debugging.[25] For a framework like TEICHI, this underlines the importance of choosing a highly fault-tolerant and open-source component design in which administrators (in our case, humanities scholars) may compose their own styles and interactions and test immediately. To this end, each component should be encapsulated and be as immune as possible to erroneous use. Provided the interfaces are well-defined and made public, the technology will then be given a chance to mature. In the end, hopefully, the frameworks will cover all or most of the life cycle of an electronic edition

or archive, from encoding, publishing, and modifying documents to contextualizing, searching, and downloading them for further use. As our comparison shows, this is not yet achieved by any of the present tools, including our own.[26]

39 We hope that TEICHI will continue to grow, offer novel features and address new application areas. The challenge is then to keep the tool simple while also offering additional features and easy customisation for advanced users. Simplicity for TEICHI means basically two things: (a) the framework must be fully functional out-of-the-box and (b) it must be easily customized. This requires integration of the tool suite into Drupal's administration GUI and its visual theme environment, which allows for easy installation and customization. Secondly, keeping the overall design simple can be achieved by organizing the tool in modules, which allows for activating and/or customizing only those modules of the tool suite that are actually needed.

40 In general, making the tool more generic and flexible will increase its applicability. The most obvious need is to broaden the tagset to cover at least TEI Lite, including some GUI-based customization features. Also, implementing more than one content type and filter could allow integrating more than one set of XSL+CSS files, thereby offering default solutions for quite specific types of projects, such as editions dealing with print, manuscript, or bilingual material. Moreover, the integration of an image viewer to include digital facsimiles will be essential for future dissemination.[27] Also on the to-do-list is an index module for terms such as place and person names, based on TEI's `<index>` and `<name>` mechanism. Finally, to further bridge the gap between encoding and display, having a TEI-aware XML editor would be a big step forward, either by supplying an API for external editors, allowing them to access the Drupal database, or by embedding an already existing online editor into Drupal.[28]

BIBLIOGRAPHY

Bérardier de Bataut, François-Joseph. 2010. *Essai sur le récit, ou entretiens sur la manière de raconter. Édition électronique.* Edited by Christof Schöch. http://www.berardier.org.

Burghart, Marjorie, and Malte Rehbein. 2011. "The Present and Future of the TEI Community for Manuscript Encoding." *Journal of the Text Encoding Initiative*, Issue 2 (February 2012). http://jtei.revues.org/372.

Burnard, Lou, and Syd Bauman. 2007. *Guidelines for Electronic Text Encoding and Interchange*, P5. Text Encoding Inititative. http://www.tei-c.org/release/doc/tei-p5-doc/en/html/index.html.

Clement, Tanya, and Susan Schreibman. 2010. "The Newest Version of the Versioning Machine (V4)." Poster presented at the 2010 Conference and Members' Meeting of the Text Encoding Initiative. Zadar, Croatia. November 8–14.

Czmiel, Alexander. 2008. "Editio ex machina – Digital Scholarly Editions out of the Box." *Digital Humanities 2008: Conference Abstracts*, edited by Lisa Lena Opas-Hänninen, Mikko Jokelainen, Ilkka Juuso, and Tapio Seppänen, 101–102. Oulu, Finland: University of Oulu. http://www.ekl.oulu.fi/dh2008/Digital%20Humanities%202008%20Book%20of%20Abstracts.pdf.

Kleist, Aaron J., ed. 2011. *Digital Ælfric Project.* http://aelfric.net/.

Freytag-Loringhoven, Baroness Elsa von. 2009. *In Transition: Selected Poems by the Baroness Elsa von Freytag-Loringhoven.* Edited by Tanya Clement. College Park, MD: University of Maryland Libraries. http://digital.lib.umd.edu/transition/.

Inscriptiones Graecae. 2011. *Inscriptiones Graecae.* Berlin: Berlin-Brandenburgische Akademie der Wissenschaften. http://telota.bbaw.de/ig/.

Humboldt, Alexander von, and Christian Gottfried Ehrenberg. 2008. *Briefwechsel.* Edited by Anne Jobst and Eberhard Knobloch. Berlin: BBAW. http://telota.bbaw.de/AvHBriefedition.

Kiernan, Kevin. 2006. "Digital Facsimile in Editing". In *Electronic Textual Editing*, edited by Lou Burnard, Katherine O'Brien O'Keeffe, and John Unsworth, 262–268. New York: MLA.

Ko, Andrew J., et al. 2011 (forthcoming). "The State of the Art in End-User Software Engineering." *ACM Computing Surveys.* doi:10.1.1.159.8597.

Kumar, Amit, Susan Schreibman, Stewart Arneil, Martin Holmes, Alejandro Bia and John Walsh. 2005. "<teiPublisher>. A Repository Management System for TEI Documents." *Literary and Linguist Computing*, 20 (1), 117–132. doi:10.1093/llc/fqh047.

Reichelt, Lisa. 2009. "Who is D7UX for?" *Drupal 7 User Experience Project.* http://www.d7ux.org/who-is-d7ux-for/.

Schlitz, Stephanie A., and Garrick S. Bodine. 2009. "The TEIViewer: Facilitating the transition from XML to web display." *Literary and Linguistic Computing*, 24 (3), 339–346. doi:10.1093/llc/fqp022.

Schreibman, Susan, Ann Hanlon, Sean Daugherty, and Tony Ross. 2007. "The Versioning Machine v3.1: A Tool for Displaying and Comparing Different Versions of Literary Texts." Paper presented at the 2007 Digital Humanities Conference. Urbana, Illinois. June 2–8.

Schreibman, Susan, and Ann M. Hanlon. 2010. "Determining Value for Digital Humanities Tools: Report on a Survey of Tool Developers." *Digital Humanities Quarterly* 4 (2).

Spaulding, Aaron, and Julie S. Weber. 2009. "Usability Engineering Methods for Interactive Intelligent Systems." *AI Magazine* 30 (4), 41–47. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2272/0.

NOTES

1. Fedora Commons Repository Software, http://fedora-commons.org/; Open Source Native XML Database eXist-db, http://exist.sourceforge.net/.

2. The Versioning Machine, v. 4.0, 2010, http://v-machine.org. See Clement and Schreibman 2010.

3. Scalable Architecture for Digital Editions, http://www.bbaw.de/telota/projekte/digitale-editionen/sade/. See Czmiel 2008 for a short presentation.

4. TEIViewer, http://teiviewer.org/. See Schlitz and Bodine 2009.

5. TEI Display plugin, http://omeka.org/codex/Plugins/TeiDisplay.

6. TEI Publisher, http://teipublisher.sourceforge.net/. See Kumar et al. 2005.

7. XML Content, http://drupal.org/project/xmlcontent.

8. For some background on Drupal's usability and development approach, see Reichelt 2009.

9. For more information on the TEICHI framework, see http://www.teichi.org/.

10. For a useful discussion of user needs analysis in software development, although applied to intelligent systems development, see Spaulding and Weber 2009.

11. We have excluded the Fedora Commons Repository Software and teiPublisher from this survey since the first is not a complete tool but rather "designed to be integrated into an application or system that provides additional functions", and the second requires more complex integration with eXist and Lucene.

12. A more substantive comparison would require two things: a preliminary survey of scholarly editors' requirements, which are not yet well-documented, and a systematic series of usability tests using a set of unified reference texts and editorial activities. Such an enquiry, deployed on each of the three publishing tools, is clearly out of the scope of this paper.

13. For how tool developers in Digital Humanities perceive measures of success of their tools, see Schreibman and Hanlon 2010, 24-26.

14. SADE is in constant development, the latest additions being a tool for linking transcribed text and corresponding parts of a facsimile as well as an interactive timeline and map feature.

15. Since all three tools compared here rely on an external editor for text encoding, we do not cover this feature here.

16. The TEICHI documentation lists the currently supported elements of TEI Lite along with information on their default attributes, display options, and suggestions for encoding.

17. Drupal Content Management System, http://drupal.org; see especially http://drupal.org/start/.

18. Drupal's documentation covers all hooks that interact with the Drupal core. See Drupal Hook API, http://api.drupal.org/api/drupal/includes--module.inc/group/hooks/7.

19. For the sake of clarity, initial error checks are omitted from figure 6. The code shown for adding CSS and Javascript is for demonstration purposes only.

20. The XSLT C library for Gnome, http://xmlsoft.org/XSLT/.

21. SAXON: The XSLT and XQuery Processor, http://saxon.sourceforge.net/.

22. eXist-db Open Source Native XML Database, http://www.exist-db.org/.

23. All components on which SADE is based are open source software; according to Alexander Czmiel from the SADE project, all custom scripts may be freely used.

24. The XSLT C library "also implements most of the EXSLT set of processor-portable extensions functions and some of Saxon's evaluate and expressions extensions", according to the project website (http://xmlsoft.org/XSLT/).

25. For a recent survey on end-user software engineering, see Ko et al. 2010.

26. The feedback we have received up to now from various researchers involved with TEI-based digital editions, generally indicating their interest in and willingness to make use of or contribute to our project, makes us optimistic that our modular, open-source approach is heading in the right direction.

27. On the importance of digital facsimiles, see for instance Kiernan 2006. An image viewer may be integrated into Drupal using tools like Seadragon Ajax, a JavaScript image viewing library (see http://gallery.expression.microsoft.com/SeadragonAjax) or Yoxview, a jQuery image viewer plugin for which a Drupal module already exists (see http://www.yoxigen.com/yoxview and http://drupal.org/project/yoxview).

28. The external editors could be Oxygen or XML Copy Editor (http://www.oxygenxml.com/ and http://xml-copy-editor.sourceforge.net/); potential online editors are Code Mirror or the XML Web GUI (http://codemirror.net/ and http://xmlwebgui.sourceforge.net/).

ABSTRACTS

This paper presents a newly developed framework for online publishing of scholarly text editions based on the TEI Guidelines. At the core of our publishing framework is support for delivery of TEI-encoded documents in Drupal, a popular, fairly powerful, and modular content management system (CMS). In this article we present the TEICHI suite of modules for Drupal that we developed, as well as a prototype implementation. TEICHI consists of a collection of modules for displaying TEI files online (via XSLT and CSS), interacting with them (via JavaScript), as well as uploading, searching, and downloading them. To showcase the particular strengths and possible use cases of our framework, we compare our tool to other currently available systems. Here, we focus on those tools which are suitable for textual scholars new to Digital Humanities and the TEI who would like to use the powerful encoding mechanisms provided by the TEI but have relatively little technical expertise. Therefore, we compare the TEICHI framework specifically to the Versioning Machine (VM) and the Scalable Architecture for Digital Editions (SADE). We look at these tools from four perspectives: that of the user interacting with the digital edition or archive, of the editor encoding and publishing texts, of the administrator setting up the publishing tool, and of the programmer possibly modifying or enhancing the tool. Our more general aim here is to investigate, from the perspective of tool development in the area of online delivery of TEI-encoded documents, what could be called the "tools paradox": there is evidence for textual scholars' need of such tools, and a number of them are available; however, the existing tools are not widely adopted by scholars. Our findings suggest that tool development has to address two aims which seem to be mutually exclusive, that of "keeping it simple" and that of "going generic". In fact, we suggest that tool developers need to find ways of turning these conflicting aims into concurrent aims if they want to build successful tools and broaden their user base.

INDEX

Keywords: Drupal CMS, online publishing tools, tool development

AUTHORS

SEBASTIAN PAPE

Sebastian Pape has obtained an M.A. in Computer Science and Mathematics from Darmstadt University of Technology and is currently finishing his PhD at the University of Kassel.

CHRISTOF SCHÖCH

Christof Schöch has obtained a Dr. phil. in French literature in 2008 from the Universities of Kassel and Paris-Sorbonne and is currently working in the DARIAH-DE project at the University of Würzburg, Germany.

LUTZ WEGNER

Lutz Wegner has obtained an M.B.A. in industrial engineering in 1974, a PhD in 1977, and his Dr. habil. degree in 1982, all from the University of Karlsruhe, Germany. In 1987 he became Associate Professor for Computer Science at the Department of Mathematics of the University of Kassel and, since 1989, holds a chair in Database Systems at the Dept. of EECS in Kassel.