USING MS WORD AS AN IMPROVISED TOOL FOR DIGITAL HUMANITIES DATA PREPARATION

Katia P. Mayfield Emanuel S. Grant Crystal Alberts
Department of Computer Science Department of English
University of North Dakota
Grand Forks, ND 58201

Abstract

With the collaboration that is being set forth between different areas of disciplines that make up the study of Digital Humanities, there are several points of concern that need to be taken into consideration. Digital Humanities concentrates on the areas of digitizing documents for archival purposes and handling of "born digital" data. There are organizations and scholars working in this field, creating standards that should be followed during data preparation, when digitizing material and making it public through online media outlet. A large effort of preparing the data correctly, besides following specific standards, depends on the availability of tools that an author may use to simplify the process. This paper focuses on presenting a report on the use of oXygen, an Extensible Markup Language (XML) editor that follows standards for Digital Humanities, and a tool developed through the use of macros in Microsoft Word to perform similar tasks.

1 INTRODUCTION

A new field that is starting to emerge in an academic setting and is rapidly spreading among a diversity of users is Digital Humanities. This is an area that concentrates in some of the traditional humanities topics--such as literature, anthropology and history--and incorporates the discipline of computational sciences. Due to the combination of disciplines that make up Digital Humanities, it is sometimes referred to as humanities computing [16]. This new area is also evolving because the opportunity for "born digital" material, which has increased with today's technology. Material characterized and labeled as being "born digital" is anything created on a computer, and meant to only be viewed on a computer and not in a hard copy format [16]. However, for all of this material to become available online there is a tedious process of data preparation that must first take place.

In particular, for documents that are "born digital", the Electronic Literature Directory (ELD) was created to provide a database for those digital documents to be stored. The users involved in submitting their work to the ELD are required to use XML encoding to tag their work. They are also asked to identify the techniques that they employed in their creation [13]. In order to maintain compatibility among the archived digital material, "born digital" data and digitized documents must follow a standard which maintain consistency within each differently authored document. For this purpose, authors and organizations working on creating "born digital" material or digitizing documents are expected to follow the Text Encoding Initiative (TEI) standards in their data preparation [17]. TEI established a standard set of tags to be used in order to make the available online material to follow a uniform pattern [17]. The development of such standards created the need for productivity tools to assist in such endeavor.

Among studies focused on productivity tools, Sautter, Bohm, and Agosti describe the importance of digitization of biosystematics literature on allowing for future searches and data mining. They also propose the use of a specific tool, GoldenGATE and compare it to regular XML editors [15].

Liu, Hu and Takeishi introduce an XML environment where part of the document can be computed, according to previously established values, using the XML tags for their identification [10]. Such calculations, described by the authors as XML dependencies, seem to be of significant value on the creation of new documents [11].

Badica and Badica describe the need and tools for information extraction of HTML pages [4] while Cechticky et al propose the use of XML tags to allow easy reutilization of software modules [7]. In this case, the XML tags are used to represent features of the software and to create families of application models. In both of these studies, the need is reasserted for the use of XML tags, which help to organize the data and infer logic relations in the document.

Among the advances in increasing the productivity on data preparation, Alkhoven discusses the training required for digitization projects [1], while Anderson and Healey report the tools usually employed to access information in digital libraries [2]. Boot proposes a new software, Editor, which allow scholars to add annotation to a work already tagged with XML entries [6]. Breure presents an editorial framework, Progenetor, designed to work as a text analyzer, and to select, gather and rearrange text fragments to generate new documents [7]. These studies show the interest and goals on correctly preparing the data for digital access.

Antonacopoulos et al published a study on the procedures used to transform historical written documents to electronic ones [3]. The study focuses on the recovery of the scanned original text and does not mention the TEI standards for digital humanities or how to handle non-written articles.

In 2008, during the panel discussion "The Next Challenge: from Easy-to-Use to Easy-to-Develop. Are You Ready?"[5]. Beringer proposed the concept of end-user development, which can be easily adopted by applying modifications to traditional office tools such as MS Word [5].

This paper discusses the use of a commercial tool in the data preparation related to important digital humanities projects and an end-user developed tool built on top of MS Word. A brief description of the contents of a digital humanities document and the basic steps involved in its preparation is presented in the next section. It is followed by the description of the use of a commercial tool named oXygen. A short introduction to the use of Visual Basic in MS Word applications is next, followed by a simple implementation of a tool that could improve the productivity of those preparing digital humanities documents. Finally, a summary of the presented material closes the paper.

2 BACKGROUND

XML and TEI are two basic standards familiar to any person in the area of digital humanities. XML is very similar to HTML, however XML is frequently used to tag information on a web page to make it easily searchable, while HTML tags usually provide tools for a web designer to specify how the information should be displayed when the page is rendered.

In Digital Humanities, even though creators are encouraged to use a standard for creating their XML files, XML is better known for not having predefined tags and for being self descriptive. In other words, creators can produce all sorts of different tags for their content creating the problem of identifying at which point their page would not be considered "standard".

One of the important standards or rule of syntax in XML is that most tags used will have what is considered an opening and a closing identifier. For example, when having a tag named "place" the writer should use an opening tag <place> and therefore a corresponding closing tag </place>. Between these two tags, the developer would place the text that is described by that tag set. In this example, the text that corresponds to the name of the "place" would be found between the two tags. Another simple example of a section of XML code is shown in Figure 1.

```
<? xml version 1.0?>

<sentence>This is an example of a sentence in XML</sentence>
```

Figure 1 XML Code Example

When developing an XML document, the developer must make sure that a Document Type Definition (DTD), or schema, and the XML document exist and are compatible. In order to develop the DTD or schema, the developer must conduct some research to obtain the descriptions of the elements and attributes that will be used. The developer must also be aware of requirements that must be met so that the appropriate element tags are created [12]. Since there is freedom for a creator to craft his own element tags, standards produced by TEI become important on the scheme of creation. Digital Humanities and the TEI standards are not new or recently created concepts. Since 1994 these standards have been recognized and used by libraries, museums, and publishers; their most important application being the preservation of historical documents, in which the TEI guidelines have had significant influence over the digitization process and the creation of Digital Libraries.

3 OXYGEN

Several tools can be used in the data preparation of Digital Humanities documents. One of the common tools used is oXygen [14]. oXygen is an XML editor developed to help those working in creating XML documents, according to different XML standards, including the Textual Encoding Initiative. This program provides four different perspectives that the user has to create or edit a document. One is a WYSIWYG XML editor, shown in Figure 2, which is used mainly by non-technical authors due to its similarities of word processing.

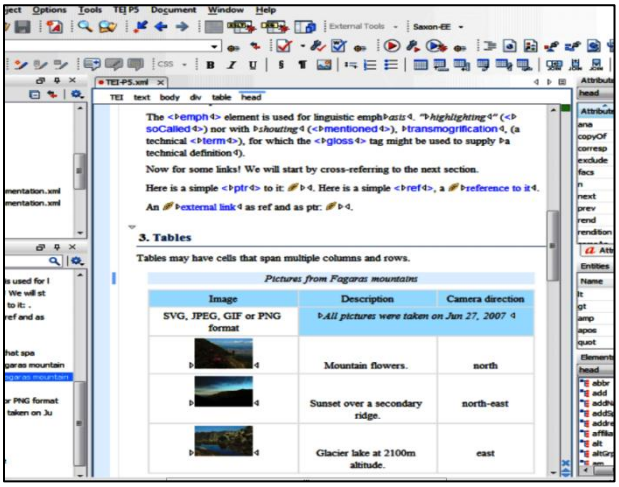

Figure 2 WYSIWYG Editor

A second approach is the format of a Grid editor which allows for repetitive XML data to be formatted in a table-like area similar to a spreadsheet. A third interface is a Tree editor, which is helpful when working with large documents so that the full document does not have to be visible, creating a "small memory footprint". Finally, there is the XML Text editor, shown in Figure 3, which is the most common perspective used for the purpose of digitizing documents.

This mode offers full control of the file to the developer, with the user being able to specify the standard to be followed. oXygen will only display tags that correspond to the standard. When the user starts to type a tag, oXygen provides a list of tags available based on the standard. At that point the user can select the tag from the menu, as shown in Figure 4, and oXygen will populate the opening and closing tag, as shown in Figure 5. The user either types information between the two tags or cuts the ending tag and pastes it in the appropriate position.

By having the user select the tag from the options displayed, the XML document is kept valid and the user does not have to be concerned with the relations between the elements. If by chance the user types something into the document that makes it invalid, a small red rectangle appears on the scroll area, which the user can choose to fix or ignore.

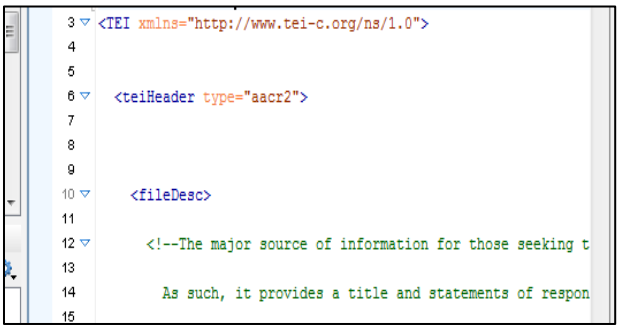

Figure 3 Text Editor

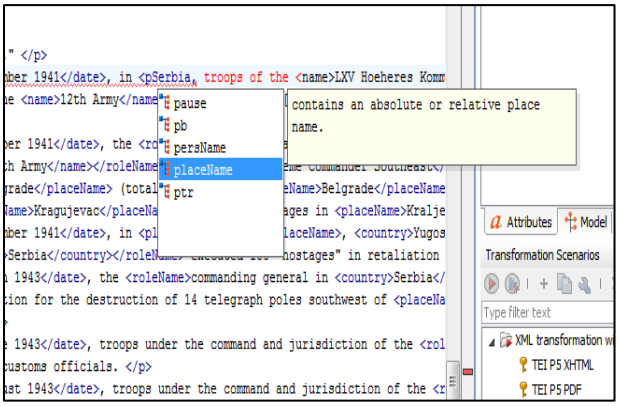

Figure 4 Text Editor Menu options for tags to use

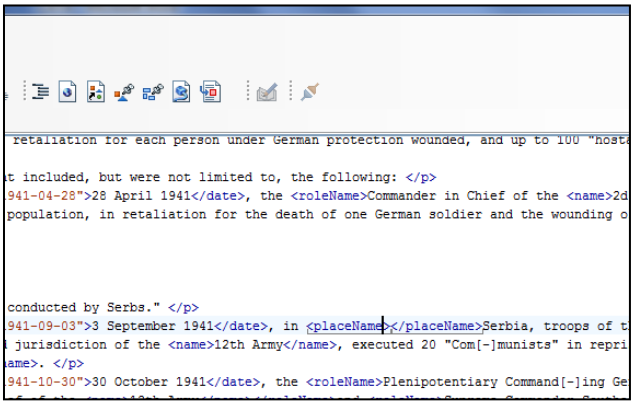

Figure 5 oXygen populates opening and closing tags next to each other

If the error is ignored, the user will later run a validation check and the error will display on the bottom portion of the screen. When the error is double clicked on, it will take the user to the appropriate line of code that needs attention.

4 IMPROVISED MS WORD TOOL

For some developers, the purchase of a product like oXygen may not be in their budget as their "born digital" or digitization projects may be of a small nature. Visual Basic (VB) is a language used to develop applications to be processed on computers running Windows operating systems. This language allows for the developer to visually design the interface of an application. The developer can create applications with graphical windows, menus, dialog boxes, work with databases and use Internet technology [9]. The Microsoft Visual Studio environment presents different interfaces for the VB developer. The developer can work in a design view, where creating the application is a graphical process, or in a code view where the developer writes the program that implements the application. These two different views make up the main window of the VB environment. A second significant window is the solution explorer window, which is used to navigate between different files in a single project. Microsoft Office products, such as Word and Excel, allow the use of VB for programming of new features not included in the original tools. This language is called VBA (Visual Basic for Applications).

With some knowledge of VBA, an user may apply an improvised solution to accomplish the digitization task. The easiest improvised way is to use Microsoft's Word program. The creator can write a Macro (VB script) to be used. For technology savvy developers that are familiar with programming, it may not be difficult to do the research on the types of elements that their XML file will need to contain and therefore he can create these as part of his VB script. Figure 6 shows the code used to create elements of "persName", "orgName", "authName", and "placeName".

The experimental tool created for this study, assumes that the user opens a new tag by clicking on a button in the Word toolbar. After the user has placed an open tag, an end tag button becomes available. In order to control such a feature, different VBA functions need to communicate. Therefore, the code includes a few public variables. Such variables are shown in the code below, where the Nametags variable stores the list of required elements. VBA allows dynamic arrays, which permit the insertion of new elements at any time by just adding their names to the variable Nametags.

```
Public index(10) As Integer
Public Nametags As Variant
Public xtag(10) As CommandBarControl
Public xtagb(10) As CommandBarControl
```

```
' Macro that builds the entire new tool bar with necessary options
Sub DigHumx()
Dim XMToolBar As CommandBar

    'Declares tags to be used...the initialization
    Nametags = Array("persName", "orgName", "authName", "placeName")

   ' Create and display the Toolbar.
    Set XMToolBar = CommandBars.Add(Name:="XMLCommonTags", _
      Position:=msoBarLeft, Temporary:=True)
    XMToolBar.Visible = True
      ' Create a open tag buttons in the Toolbar (repeated for each tag).
    For i = 0 To UBound(Nametags)
        Set xtag(i) = XMToolBar.Controls.Add(Type:=msoControlButton)
        xtag(i).Visible = True
        xtag(i).Style = msoButtonCaption
        xtag(i).Caption = "| <" & Nametags(i) & "> |"
        xtag(i).Tag = i
        xtag(i).OnAction = "ButtonAction0"
     Next i

   ' Create a open tag button in the Toolbar for closing tags, initially invisible.
    For j = 0 To UBound(Nametags)
        Set xtagb(j) = XMToolBar.Controls.Add(Type:=msoControlButton)
        xtagb(j).Visible = False
        xtagb(j).Style = msoButtonCaption
        xtagb(j).Caption = "| </" & Nametags(j) & "> |"
        xtagb(j).Tag = j
        xtagb(j).OnAction = "ButtonAction5"
    Next j

End Sub
```

Figure 6. Main code used to create toolbar

The main macro, shown in Figure 6, is executed once by the user. Its function is to add the required buttons to the

toolbar. It consists of three sections of instructions. The first is responsible for the creation of the new toolbar. The second and third are loops used to set up the buttons to be clicked by the user. The first loop creates the open tags and makes them visible, while the second loop creates the close tags, hiding them from the user. Those loops also determine the functions to be executed when a button is clicked. The functions are shown in Figure 7.

```
Sub ButtonAction0()
    current = CommandBars.ActionControl.Tag
    'index keeps track of how many tags are open
    index(current) = index(current) + 1
    'turns the closing tag visible
    xtagb(current).Visible = True
    Selection.TypeText ("<" & Nametags(current) & ">")
End Sub

Sub ButtonAction5()
    current = CommandBars.ActionControl.Tag
    Selection.TypeText ("</" & Nametags(current) &
">")
    index(current) = index(current) - 1
    If index(current) = 0 Then
        xtagb(current).Visible = False
    End If
End Sub
```

Figure 7 Code for functions that control actions of buttons

Function ButtonAction0 inserts an opening tag in the text while incrementing the variable index to record how many of these tags are still open (in the case of nested tags) and changes the visibility of the corresponding closing tag button, making it available to the user. Function ButtonAction5, activated when the user clicks a closing tag, decrements the open tags counter (index) and if the result is zero, it changes its visibility to "erase" the button from the toolbar.

From the user point of view, the subroutine ButtonAction0 displays the elements as buttons in the word document toolbar, as shown in Figure 8, so that when the creator comes across text that must be tagged, he places the cursor in the correct spot and uses the mouse to click the button. Closing tag buttons are shown in Figure 9. Once the corresponding closing tag is used within the text its button disappears, helping the creator to keep track of which elements are still open and must be closed. This is a very simple trick implemented through the use of VBA, where the buttons are created in a single execution of the main routine, while the action functions just need to change their visibility (display characteristics).

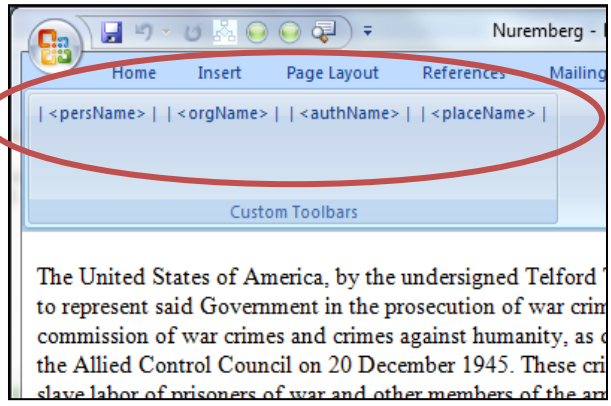

Figure 8 Buttons shown for opening tags

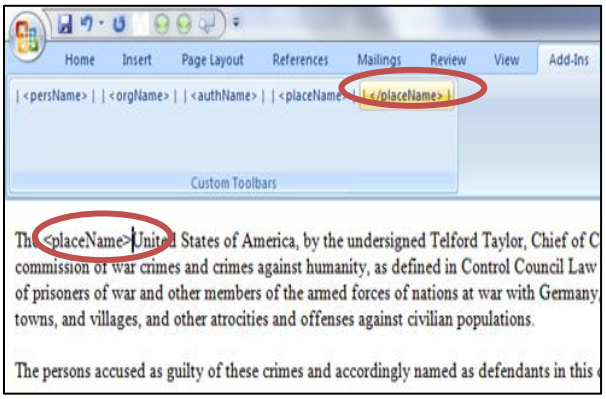

Figure 9 Opening tag in text and closing tag button

Even though MS Word does not keep the text validated as the user works in the file, it is an alternative for those working on smaller digitization and creation projects where the validation required is minimal. Figure 10 shows some of the most used tags in a digitization project in progress at the University of North Dakota [11]. A simple performance comparison between oXygen and the MS Word tool can show that in trying to insert the tag <persName>, the user, in oXygen, needs to use two key strokes in order to get the tag recognized and then a mouse click to add the opening and closing tags. After that, the user still needs to move the closing tag to its correct placement in the text, or if the user is knowledgeable on the oXygen features then the text can be highlighted before selecting the tag. While the MS Word solution is very poor in terms of features, it still needs just two mouse clicks to add a tag (one for the opening and the second for the closing).

File Name	Total Tags	pers Name	placeName
H1_20	1364	77	51
H21_41	1598	217	225
H42_83	2007	154	137
H4_95	694	49	80
H96_108	677	46	14

Figure 10 Statistical Information for Nuremberg Trial Transcript Procedures [11]

When comparing the two tools with respect to the insertion of the tag persName in the file H21_41 (Figure 10) the oXygen program would require 434 key strokes, followed by 217 mouse clicks and the adjustment of the corresponding closing tag. The MS Word solution performs the same task with 434 mouse clicks and no other action (besides the correct positioning of the cursor, required in both cases) .

5 SUMMARY

As shown in this paper, there are products already in use that are compatible with digitization standards. These products can be a very good investment for those who have large projects in progress. However, for authors who are creating their own "born digital" material or digitizing a very small collection of hard copy material, such an investment may not suit their budget due to the length of time that the software will be of use. In these cases, this paper has shown that the research usually performed by a creator to initialize a DTD file should provide enough information to allow a modest Visual Basic script to run on Microsoft Word and achieve a similar outcome as an author who uses a specialized program. Such procedure has been shown to have similar advantage to the productivity of the author, with the disadvantage of not providing the opportunity to validate the resulting product according to the standards from the Textual Encoding Initiative or any other source.

6 REFERENCES

[1] Alkhoven, P., "Digitizing cultural heritage collections: The importance of training," AHC Proceedings, 2005.

[2] Anderson, D., Healey, R.G., "Broadening the scope of electronic book publishing: A comparison of database-driven public domain software approaches of database-driven E-sources," AHC Proceedings, 2005.

[3] Antonacopoulos , A., Karatzas, D., Krawczyk, H., and Wiszniewski, B. "The Lifecycle of a Digital Historical Document: Structure and Content," DocEng '04, pp. 147-154, 2004.

[4] Badica, C. and Badica A., "Logic Wrappers and XSLT Transformations for Tuples Extraction from HTML," XSym 2005, LNCS 371, pp. 177-191, 2005.

[5] Beringer J., Fischer, G., Mussio, P. Myers, B. Patem, F., and Ruyter, B. "The next challenge: From easy-to-use to easy-to-develop. Are you ready?," CHI '08, pp. 2257-2260, Florence, 2008.

[6] Boot, P., "Advancing digital scholarship using EDITOR," AHC Proceedings, 2005.

[7] Breure, L., "PROGENETOR: An editorial framework for reuse of XML content," AHC Proceedings, 2005.

[8] Cechticky V, Pasetti A, Rohlik O, Schaufelberger W. 2004. "XML-based feature modeling," ICSR '04, pp. 101–114, 2004.

[9] Gaddis, T., and Irvine, K., "Starting out with Visual Basic 2010," 5th ed., Addison-Wesley, Indianapolis, 2011.

[10] Liu, D., Hu, Z., Takeichi, M., "An Environment for Maintaining Computation Dependency in XML Documents," DocEng '05, 2005.

[11] Mayfield, K., Grant, E., Alberts, C., "Data for a Digital Humanities document: Challenges and observations," Caine '11, 2011.

[12] Mercer, D. XML: A Beginner's Guide, McGraw-Hill, Chicago, 2001.

[13] National Endowment for the Humanities, Electronic Literature Directory. http://directory.eliterature.org/. June 29, 2011.

[14] <oXygen/> XML Editor. http://www.oxygenxml.com/ June 29, 2011.

[15] Sautter, G., Bohm, K., and Agosti, D. "Semi-automated XML markup of biosystematic legacy literature with the goldengate editor," Pacific Symposium on Biocomputing, pp. 391-402, 2007.

[16] Schreibman, S., Siemens, R., Unsworth, J. "A companion to Digital Humanities," Oxford: Blackwell, 2004.

[17] Text Encoding Initiative. http://www.tei-c.org/index.xml. June 29, 2011.