xTagger: a New Approach to Authoring Document-centric XML

Ionut E. Iacob and Alex Dekhtyar

ABSTRACT

The process of authoring document-centric XML documents in humanities disciplines is very different from the approach espoused by the standard XML editing software with the data-centric view of XML. Where data-centric XML is generated by first describing a tree structure of the encoding and then providing the content for the leaf elements, document-centric encodings start with content which is then marked up. In the paper we describe our approach to authoring document-centric XML documents and the tool, xTagger, originally developed for this purpose within the Electronic Boethius project [2], otherwise enhanced within the ARCHway project [5], an interdisciplinary project devoted to development of methods and software for preparation of image-based electronic editions of historic manuscripts.

1. INTRODUCTION

Two types of XML documents emerge from applications: *data-centric* and *document-centric*. Data-centric XML is characterized by a fairly regular structure. It occurs in the context of structured data exchange and representation of semistructured data. Document-centric XML has, in general, a much more irregular structure, is

often characterized by the ubiquitous nature of mixed markup in it and is often encountered as the means of encoding information about documents. The distinction between these two classes of XML documents becomes clear when authoring is considered. Data-centric XML is mostly computer-generated. When humans do author data-centric XML documents they typically first define the structure of the data and then “fill in the blanks” by providing the content for the leaf elements in the structure. Document-centric XML documents are much more often human-made. Typically, their creation *starts with the existing content*, and the markup is introduced “on top” of it (During the editing process the content may change too). In addition, a lot of document-centric markup is *multi-hierarchical* [6, 4]. Such different processes suggest different XML authoring software for different types of XML documents. Indeed, Clark [1] points out that XML editors can be classified as *text editors* and *structural editors*. The majority of existing XML editors fall in the second category and are used primarily to author data-centric XML documents.

Our work on building methodology and software for production of image-based electronic editions (IBEEs) of historic manuscripts [5] exposed the need for document-centric XML authoring software attuned to the humanities scholars. To address this need, we have developed xTagger, a document-centric XML editor. This editor is part of the Edition Production

Technology (EPT) and can be used both in a standalone mode and integrated with the other IBEE authoring tools. In the rest of this paper we briefly discuss the requirements for a document-centric editor (Section 2), xTagger architecture (Section 3) and its implementation (Section 4).

2. REQUIREMENTS

High-level requirements for a document-centric XML editor originate from two main sources: the needs of the intended users (in our case – humanities scholars and/or document experts), and the need to produce valid XML documents.

The key requirements obtained from the intended users are: (a) the software must be able to hide the XML syntax from the user; (b) the software must support introduction of markup by highlighting the region of the text and selecting the XML element; (c) the software must support prolonged editorial process; (d) the software must support changes in the underlying DTD/XML Schema of the encoding over time and (e) the software must support the use of multiple XML hierarchies in the editorial process.

The need to produce a valid document implies the need to closely monitor all the markup modifications introduced by the user into the document during the editorial process. In particular, whenever a new markup is introduced, the software must check the the current XML document *has a valid extension*, i.e., can become valid if more markup is introduced.

3. ARCHITECTURE

xTagger has a three-tier architecture: the data structure layer, the mediator layer, and the editor's GUI. This allows us to decouple the data management issues from the GUI and alter the internal representation of XML without changes in the top layer.

Data structure layer. This layer contains the data-structure representation of the XML document and the API for data manipulation (a document model). For instance, it is very common to represent an XML document as a in-memory DOM tree[2]. The DOM data structures allows document traversal and updates, and provides support for XML query languages[3].

Mediator layer. This layer connects the XML document representation to the document view the user interacts with. The mediator has two main functions: selects parts of the XML document (based on user's choice) to be presented in the editor's GUI, and maintains a mapping between the XML document components (for instance, DOM nodes) and the editor's GUI components (for instance, ranges over the string of characters displayed in the Editor's GUI). The mediator is also responsible for translating possible search results over the XML data structure representation into the corresponding visual components in the Editor's GUI (using data structure – Editor's GUI mapping).

Editor's GUI. Is the visible part (through mediator's filtering) of the XML document. As pointed out earlier, based on user's experience or needs, the editor's GUI may show only the content of the document or as many XML tags as the user selects for display. Markup insertion is performed by highlighting the appropriate content range, selecting the element tag (or element name, if tags names are hidden from the user) and filling in the attribute values through a schema-based template. The mediator translates the content range into a data structure range, performs the necessary document integrity checks, and eventually inserts the markup in the XML data structure. We emphasize that using validity checks is inappropriate while the XML document being edited — most of the time an incomplete document-centric XML will not be valid. Instead the more appropriate concept of *potential validity* of an XML document [3], the means of checking whether or not the document is valid *or can become valid after inserting some markup*.

4. IMPLEMENTATION

xTagger (Figure 1) is implemented in Java on top of the Eclipse platform and is designed to support multi-hierarchical XML documents based on the GODDAG [6] data structure representation[4].

The main features of xTagger are: (a) powerful filtering (it supports XML filtering as well as text filtering – only selected tag content is displayed); (b) syntax coloring for XML display and highlighting of text regions marked up with a selected tag; (c) mouse-over tooltips to indicate markup that covers a certain content range; (d) support for XPath queries over multiple XML hierarchies [4]; (e) potential validity checks for XML document [3] after document updates; (f) support for markup element and attribute aliases; (g) on the fly HTML presentations based on XSLT. In Figure 1 we show two projections of an XML document, a text content projection and an XML projection, together with a mouse-over tooltip. We note that in xTagger both text and markup insertion (tagging) can be performed in both XML projection and content projections, by selecting the appropriate content range. Also, content updates (insert/delete/cut/paste) are permitted for content and/or XML projections.

5. CONCLUSIONS

Currently available XML authoring tools rely either on schema-based templates or on the ability of the user to handle the angle brackets of the XML language. Although providing a lot of features for preserving document's integrity, the editors for document-centric XML still put on user the burden of dealing with XML syntax. In contrast, xTagger allows gives the users the ability to select the appropriate level of interaction with XML. Our current implementation is based on the in-memory GODDAG data structure, we are studying the possibility of using persistent storage (a database) in order to edit larger documents.

Acknowledgements. The authors would like to thank Kevin Kiernan for useful suggestions about the desired features of a document-centric XML authoring tool, especially in the context of working with humanities students and untrained scholars in general. We would also like to thank Dorothy C. Porter for useful comments, and Jerzy W. Jaromczyk all the students working on the ARCHway project for useful discussions.

6. REFERENCES

[1] J. Clark. Incremental XML Parsing and Validation in a Text Editor, December 2003. Presentation at XML 2003, Philadelphia.

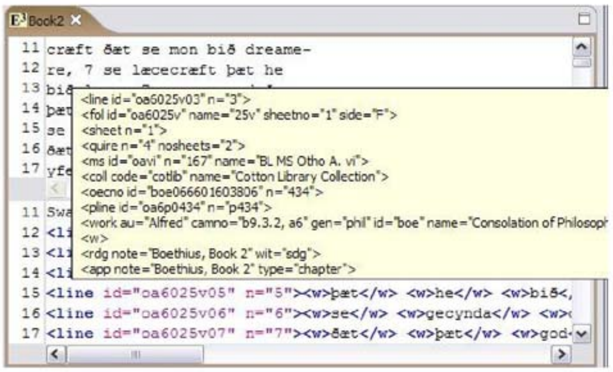
[2] K. Hawley and K. Kiernan. An Image-Based Electronic Edition of Alfred the Great's Old English Version of Boethius's Consolation of Philosophy. In *Proc. Joint International ALLS-ACH Conference*, 2003.

[3] I. E. Iacob, A. Dekhtyar, and M.I. Dekhtyar. Checking Potential Validity of XML Documents. In *In Proc. of the Seventh International Workshop on the Web and Databases (WebDB)*, pages 91–96, 2004.

[4] I. E. Iacob, A. Dekhtyar, and W. Zhao. XPath Extension for Querying Concurrent XML Markup. Technical Report TR 394-04, University of Kentucky, Department of Computer Science, February 2004. http://www.cs.uky.edu/—dekhtyar/publications/TR394-04.ps.

[5] K. Kiernan, J. Jaromczyk, A. Dekhtyar, D. Porter, K. Hawley, S. Bodapati, and I. Iacob. The ARCHway project: Architecture for research in computing for humanities through research, teaching, and learning. *Literary and Linguistic Computing*, 2004. forthcoming.

[6] C. M. Sperberg-McQueen and C. Huitfeldt. GODDAG: A Data Structure for Overlapping Hierarchies. In *Principles ofDigital Document Processing, DDEP/PODDP 2000, Munich*, pages 139–160, Sept. 2000. Early draft presented at the ACH-ALLC Conference in Charlottesville, June 1999.



```
cræft ðæt se mon bið dreame-
re, 7 se læcecræft þæt he
```

```
<line id="oa6025v03" n="3">
<fol id="oa6025v" name="25v" sheetno="1" side="F">
<sheet n="1">
<quire n="4" nosheets="2">
<ms id="oavi" n="167" name="BL MS Otho A. vi">
<coll code="cotlib" name="Cotton Library Collection">
<oecno id="boe066601603806" n="434">
<pline id="oa6p0434" n="p434">
<work au="Alfred" camno="b9.3.2, a6" gen="phil" id="boe" name="Consolation of Philosoph
<w>
<rdg note="Boethius, Book 2" wit="sdg">
<app note="Boethius, Book 2" type="chapter">
```

```
<line id="oa6025v05" n="5"><w>þæt</w> <w>he</w> <w>bið<,
<line id="oa6025v06" n="6"><w>se</w> <w>gecynda</w> <w>
<line id="oa6025v07" n="7"><w>ðæt</w> <w>þæt</w> <w>god-
```

Figure 1: The xTagger XML editor