A Framework for Processing Complex Document-centric XML with Overlapping Structures

Ionut E. Iacob*
Department of Computer Science
University of Kentucky
Lexington, KY, USA
eiaco0@cs.uky.edu

Alex Dekhtyar†
Department of Computer Science
University of Kentucky
Lexington, KY, USA
dekhtyar@cs.uky.edu

ABSTRACT

Management of multihierarchical XML encodings has attracted attention of a number of researchers both in databases [8] and in humanities[10]. Encoding documents using multiple hierarchies can yield overlapping markup. Previously proposed solutions to management of document-centric XML with overlapping markup rely on the XML expertise of humans and their ability to maintain correct schemas for complex markup languages.

We demonstrate a unified solution for management of complex, multihierarchical document-centric XML. Our framework includes software for storing, parsing, in-memory access, editing and querying, multihierarchical XML documents with conflicting structures.

1. INTRODUCTION

The popularity of XML for encoding documents is, in large part, due to four key factors: (i) its simplicity, (ii) its flexibility, (iii) open standards and (iv) availability of software tools to process raw XML data and to interface with programs that use the data.

As pointed out recently, for certain encoding tasks a single encoding hierarchy is not enough. For data-centric XML, a novel data structure for representing multihierarchical encodings was proposed by Jagadish et al. [8]. In the context of document-centric XML with overlapping structures, the term "concurrent XML" or "multihierarchical XML document" refers to the encoding of the same content with markup from more than one DTD/XML Schema. The problem of overlapping structures in multihierarchical XML documents has attracted attention of a number of humanities researchers in recent years [10, 9, 3]. Two observations can be made about multihierarchical XML documents: (a) traditional methods of XML processing often are inadequate for them, and (b) the tasks to be performed with multihierarchical XML are the same as with regular XML: parsing, in-memory storage, in-memory access, querying, authoring, maintenance, persistent storage and indexing, and querying in presistent storage.

*Work supported, in part, by the NEH grant RZ-20887-02.

†Work supported, in part, by the NSF grant ITR-0219924 and the NSF grant ITR-0325063.

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA

<r><line>gesceaftum unawendendne sin</line><line>gallice sibbe gecynde þa</line></r>

<r><vline><w>gesceaftum</w><w>unawendendne</w></vline><vline><w>singallice</w><w>sibbe</w><w>gecynde</w></vline><vline><w>þa</w></vline></r>

<r><res>gesceaftum una</res>wendendne s<res>in</res><res>gallice sibbe gecyn</res>de þa</r>

<r>gesceaftum una<dmg>w</dmg>endendne singallice sibbe gecyn<dmg>de þa</dmg></r>

Figure 1: Document encodings for an Old English manuscript.

In general, management of multihierarchical document-centric XML presents unique challenges, not found when dealing with data-centric XML. In this demo, we present our approach to this problem. At present, we have addressed the issues related to parsing, representation and storage, and querying document-centric multihierarchical XML documents in main memory. Work on building persistant storage solutions is currently underway. In Section 2 we describe the notion of concurrent XML hierarchies. In Section 3 we briefly outline the framework for representing concurrent XML in main memory (for a detailed description of this framework see [2, 6]. Finally, in Section 4 we give a brief outline of the demo and the software components we will be presenting.

2. CONCURRENT HIERARCHIES

The problem of overlapping markup has been known to the text encoding community and the humanities scholars since the days of SGML [9]. Recently, with the switch of Text Encoding Initiative (TEI) Guidelines [10] from SGML to XML, there has been a re-emergence of interest to this problem in the context of XML encodings as evidenced by [10, 3, 12]. This problem typically manifests itself when a researcher must encode in XML a wide variety of features for a large document (e.g., a book or a manuscript). Some of the features may form so called *concurrent hierarchies*.

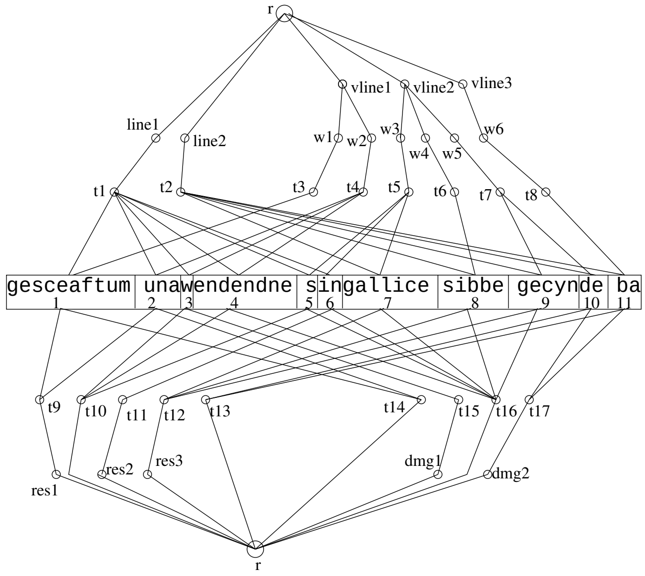

Figure 2: The GODDAG data structure for the text in Figure 1

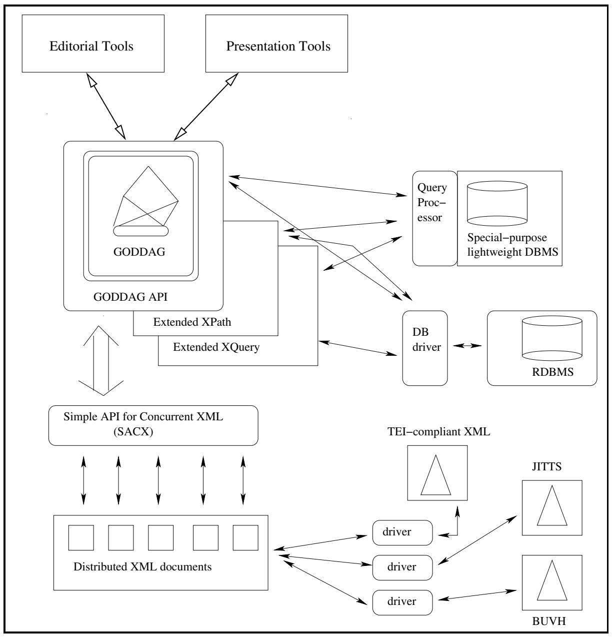

Figure 3: A framework for management of Concurrent XML Hierarchies

A hierarchy is formed by a subset of the elements of the markup language used to encode the document. The elements within a hierarchy have a clear nested structure. When more than one such hierarchy is present in the markup language, the hierarchies are called concurrent.

A typical example of concurrent hierarchies is the XML markup used to encode the physical location of text in a printed edition: book, page, physical line, vs. the markup used to encode linguistic information about the text: sentence, phrase, word, letter. The key problem with using concurrent hierarchies to encode documents is that *markup in one hierarchy is not necessarily well-formed with respect to the markup in another hierarchy*.

As an example, we consider the encoding of an Old English manuscript [1]. The problem can be stated as follows. Given images of manuscript folios, one wants to encode on the content of the manuscript a variety of document features: manuscript physical structure (lines, pages), document structure (words, sentences, verses), text restorations, manuscript damages, etc.

In Figure 1 a fragment of the manuscript is presented, together with four desired encodings on the top of the manuscript fragment's content. All documents have the same root element tag `<r>`. It is not difficult to see that some encodings are conflicting with each other (for instance, some of `<w>` markup are in conflict with `<line>`, `<res>`, or `<dmg>`), so a single XML representation is not straightforward possible (a non well-formed document would result).

To store concurrent markup in a single XML document, the exact representation of the document features has to be sacrificed in order to preserve the well-formedness of the document. TEI Guidelines [10] suggest two solutions: *fragmentation* (fragment markup ranges where overlapping and "glue" fragments together using a special attribute carrying the same ID for all fragments) and *milestones* (declare elements that are likely to produce overlapping as empty elements). Using these solutions it is possible to put concurrent markup in the single XML document, however, the underlying semantics of the markup and the DOM tree semantics of the XML document will differ. In particular, this makes querying such XML documents a complicated task.

This work attempts to bridge the gap between the apparent necessity for concurrent markup and the lack of software support for it by proposing a framework for the creation, storage, maintenance, transforming, and querying the concurrent XML markup.

3. A FRAMEWORK FOR MANAGEMENT OF CONCURRENT MARKUP HIERARCHIES

The main idea of our approach is to group non conflicting tag elements into separate DTDs (schemas). More formally, we define a *concurrent markup hierarchy* as a collection of DTD elements that are not in conflict to each-other.

DEFINITION 1. [2] *A* concurrent markup hierarchy CMH *is a tuple* $CMH = \langle \rho, \{T_1, T_2, ..., T_k\} \rangle$ *where:*
• ρ *is an XML element called the* root *of the hierarchy;*
• $T_i,\ i = \overline{1, k}$ *are DTDs such that:*
(i) $\forall 1 \leq j \leq k, i \neq j,\ elements(T_i) \cap elements(T_j) = \{\rho\}$*;*
(ii) $\forall t \in elements(T_i) - \{\rho\}$*,* ρ *is an ancestor of* t *in* T_i*.*

Based on a concurrent markup hierarchy CMH, we define a *virtual* union of XML documents (one document corresponds to a DTD in CMH) that have the same content, the same root element, and that are encoded with elements from the corresponding DTD.

DEFINITION 2. [2] *A* distributed XML document D *over a concurrent markup hierarchy* $CMH = \langle \rho, \{T_1, T_2, ..., T_k\} \rangle$ *is a collection of XML documents:* $D = \langle d_1, \ldots, d_k \rangle$ *where (i)* $(\forall 1 \leq i \leq k)$ d_i *is valid w.r.t.* T_i*; (ii)* $string(root(d_1)) = string(root(d_2)) = ... = string(root(d_k))$*, and (iii)* $root(d_1) = root(d_2) = ... = root(d_k) = \rho$*.*

We say that for a distributed document D*,* $string(root(D)) = string(root(d_1))$ *and* $root(D) = root(d_1)$*.*

An example of distributed document is the set of four document encodings in Figure 1. To represent all hierarchy encodings as a single data structure, we first divide the document content into *leaf nodes* (fragments), where the borders are given by markup positions from all hierarchies (we call these text divisions *leaves*). Each markup structure is represented as an extended DOM tree (where text nodes have leaves as children), then all trees are united at the root (the same root for all trees) and at the leaf level (text nodes are the same). It results in a directed graph structure: Generalized Ordered Descendant Direct Acyclic Graph (GODDAG) [11]. For instance, the GODDAG of the document with overlapping structure in Figure 1 is given in Figure 2 (the numeric labels next to tag labels are just for the purpose of identification). This structure is a natural extension of DOM trees. It is easy to navigate it and execute pattern queries on it. We note that navigation from one structure to another is done through root node or leaf (text) nodes.

The framework we propose (Figure 3) for management of concurrent hierarchies generalizes the traditional XML processing framework (Figure 3). We need to address the following problems: (i) parsing concurrent XML document into a GODDAG structure (or, alternatively, constructing a GODDAG from an XML database), (ii) develop DOM-style API to access and navigate GODDAG and (iii) develop the semantics and algorithms for a path expression language (XPath) over GODDAG. In addition, we need to provide support for authoring document-centric multihierarchical XML documents. On top of the GODDAG data structure we have defined and implemented a path query language (an extension of the XPath query language; an XQuery extension and implementation is under development). We have also defined the means of building GODDAGs from distributed XML documents via a parser for concurrent XML (SACX [6]). Distributed XML documents can be created directly using the authoring tools we developed[4], or can be obtained, via drivers, from a variety of proposed representations of concurrent XML markup[2]. The GODDAG API we define and implement allows top level user tools (authoring or presentation tools) to query and update multihierarchical XML documents represented by GODDAG.

4. DEMONSTRATION OVERVIEW

The following problems are addressed in our framework:

• *Data model for concurrent document-centric XML.* We use the GODDAG data structure [11], a generalization of DOM trees for XML, to represent multihierarchical XML documents. Our demo includes a DOM-like API for the GODDAG data structure and a GODDAG parser from a variety of XML representations of multihierarchical documents [6].

• *Querying concurrent XML.* XPath and XQuery are inefficient in expressing certain important information needs over concurrent XML documents (e.g., requests for overlapping content given two tags). In addition, XPath is defined on the DOM Tree structure, whereas concurrent XML documents are modelled using GODDAG graphs. We redefine the XPath semantics on GODDAG (we call it the Extended XPath), and extend it with features that are specific to processing of concurrent XML [7], such as the `overlapping` axis. We show an efficient implementation of the Extended XPath and how it can be used to answer meaninful queries in the context of multihierarchical XML.

• *Authoring tools for document-centric XML.* Our software suite includes xTagger, a specialized editor for multihierarchical document-centric XML. xTagger allows users to select a document fragment and choose the approriate markup for it (from any of the XML hierarchies associated with the document). It implements prevalidation checking, which detects encodings that cannot be extended to valid XML with further markup insertions[5].

• *Document manipulation.* One of the advantages of the proposed framework is its flexibility: concurrent XML can be imported into/exported from our software suite from/to a wide range of representations ([2]). We demo the filtering feature for partially viewing and/or exporting a subset of document encodings.

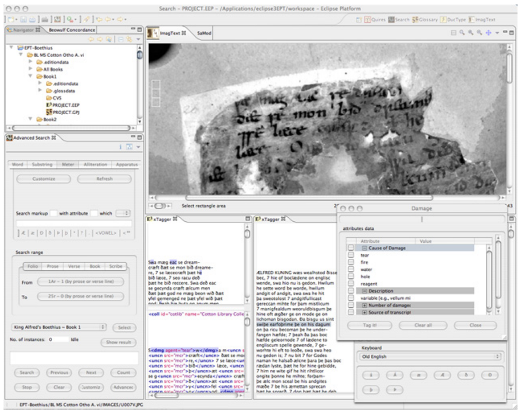

Figure 4: The EPT overview

The sofware we demonstate is part of the Edition Production Technology (EPT, Figure 4) toolkit for creating Image-based electronic editions of historic manuscripts created at the University of Kentucky. The software is implemented in Java on Eclipse platform. We demonstrate the software using the electronic edition of the Xth century Old English manuscript of Boethius' "Consolation of Philosophy."

5. REFERENCES

[1] A. M. S. Boethius. Consolation of philosophy. *Alfred The Great (translator), British Library MS Cotton Otho A. vi*. Manuscript, folio 36v.

[2] A. Dekhtyar and I. E. Iacob. A Framework for Management of Concurrent XML Markup. *Data and Knowledge Engineering*, 52(2):185–215, 2005.

[3] P. Durusau and M. B. O'Donnell. Concurrent Markup for XML Documents. In *Proc. XML Europe*, May 2002.

[4] I. E. Iacob and A. Dekhtyar. xtagger: a new approach to authoring document-centric xml. In *Proc. Joint Conference on Digital Libraries*, 2005. accepted.

[5] 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.

[6] I. E. Iacob, A. Dekhtyar, and K. Kaneko. Parsing Concurrent XML. In *Proceedings WIDM*, pages 23–30, November 2004.

[7] 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.

[8] H. V. Jagadish, L. V. S. Lakshmanan, M. Scannapieco, D. Srivastava, and N. Wiwatwattana. Colorful xml: one hierarchy isn't enough. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 251–262. ACM Press, 2004.

[9] A. Renear, E. Mylonas, and D. Durand. Refining our Notion of What Text Really Is: The Problem of Overlapping Hierarchies. *Research in Humanities Computing*, 1993. N. Ide and S. Hockey, (Eds.).

[10] C. M. Sperberg-McQueen and L. Burnard(Eds.). Guidelines for Text Encoding and Interchange (P4). http://www.tei-c.org/P4X/index.html, 2001. The TEI Consortium.

[11] C. M. Sperberg-McQueen and C. Huitfeldt. GODDAG: A Data Structure for Overlapping Hierarchies. In *DDEP/PODDP*, pages 139–160, Sept. 2000.

[12] A. Witt. Meaning and interpretation of concurrent markup. In *Proc., Joint Conference of the ALLC and ACH*, pages 145–147, 2002.