

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

The key of overlapping structures or *concurrent markup hierarchies* in XML encodings of documents is that markup in one hierarchy is not necessarily well-formed with respect to the markup in another hierarchy. Previously proposed solutions to this problem rely on the XML expertise of humans and their ability to maintain correct schemas for complex markup languages.

We demonstrate a 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

XML document encoding's popularity is 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, sometimes a single encoding hierarchy is not enough and a novel data structure for representing data-centric XML in the presence of several hierarchies was proposed [7]. 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. 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 persistent 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA
Copyright 2005 ACM 1-59593-060-4/05/06 ...\$5.00.



```
<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.

Despite the fact that a lot of research has been conducted in database communities on storing, querying, or authoring XML documents, managing document-centric XML from humanities applications have captured little attention from computer scientists. Here, we demonstrate our approach to the problem of management of multihierarchical XML in main memory structure (research on storing and querying it in persistent storage is currently underway).

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 [8]. Recently, with the switch of Text Encoding Initiative (TEI) Guidelines [9] from SGML to XML, there has been a re-emergence of interest to this problem in the context of XML encodings as evidenced by [9, 3, 11]. 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*. 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:

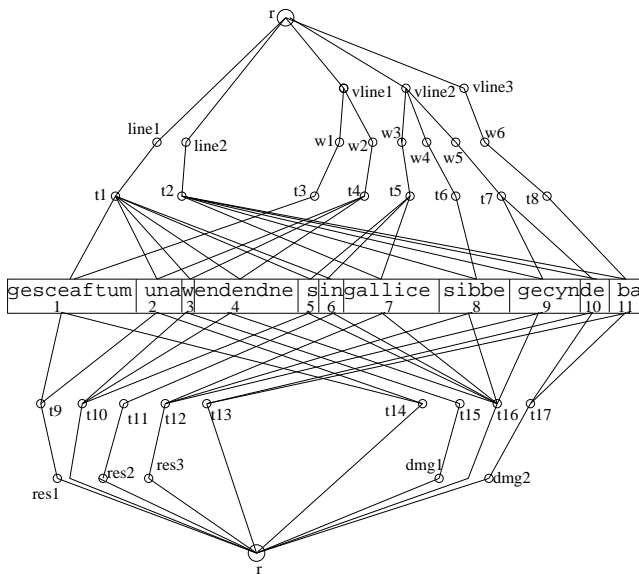


Figure 2: The GODDAG data structure for the text in Figure 1

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 $\langle r \rangle$. It is not difficult to see that some encodings are conflicting with each other (for instance, some of $\langle w \rangle$ markup are in conflict with $\langle line \rangle$, $\langle res \rangle$, or $\langle dmg \rangle$), so a single XML representation is not straightforward possible (a non well-formed document would result).

Concurrent markup lessen the attractiveness of XML as the following dilemma arise: exact document features encoding versus XML representation. Humanities researchers [9] came up with two quick solutions for the problem matter: 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). It is not difficult to notice that, although the solutions above are solving an *XML representation* problem for overlapping hierarchies, they increase the burden of maintaining such documents and makes processing (especially querying and transforming) a very expensive 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.

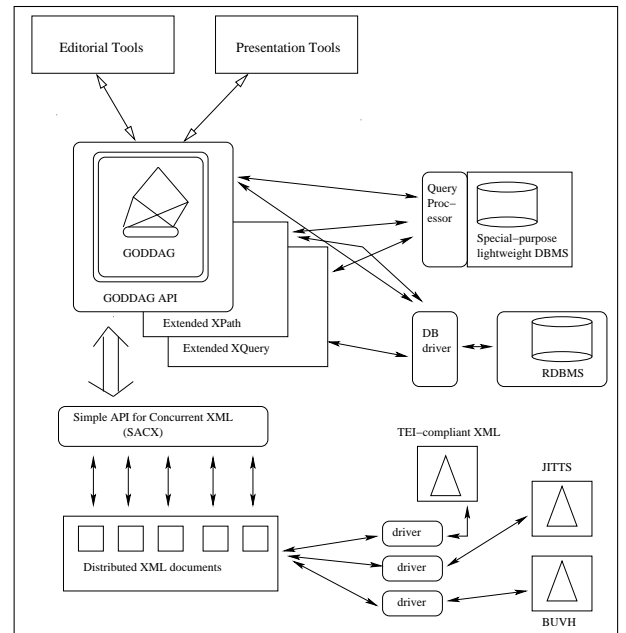


Figure 3: A framework for management of Concurrent XML Hierarchies

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. 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 $CMH = \langle \rho, \{T_1, T_2, \dots, T_k\} \rangle$ where:

- ρ is an XML element called the root of the hierarchy;
- $T_i, i = 1, k$ are DTDs such that:
 - $\forall 1 \leq j \leq k, i \neq j, elements(T_i) \cap elements(T_j) = \{\rho\}$;
 - $\forall t \in elements(T_i) - \{\rho\}, \rho$ 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 $CMH = \langle \rho, \{T_1, T_2, \dots, T_k\} \rangle$ is a collection of XML documents: $D = \langle d_1, \dots, d_k \rangle$ where (i) $(\forall 1 \leq i \leq k) d_i$ is valid w.r.t. T_i ; (ii) $string(root(d_1)) = string(root(d_2)) = \dots = string(root(d_k))$, and (iii) $root(d_1) = root(d_2) = \dots = root(d_k) = \rho$.

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 text division, 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 a directed graph structure: Generalized Ordered Descendant Direct Acyclic Graph (GODDAG [10]). 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). The main advantages of using such a data structure stem from the tree-like data structure: easy to navigate and execute pattern queries. 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: parsing XML document into a DOM data structure (or, alternatively, constructing DOM from an XML database), then use the DOM API support for editing, querying, and transforming the XML document. In this framework, we use a GODDAG data structure as a multihierarchical XML document model and we defined and implemented a query language (an extension of the XPath query language; an XQuery extension and implementation is under development) for searching this in-memory data structure. A GODDAG can be constructed from a distributed XML document via a parser for concurrent XML (SACX [5]) or, alternatively, from a persistent storage (relational database or specialized database). The distributed XML document can be directly obtained, via drivers, from a variety of proposed representations of concurrent XML markup: TEI's fragmentation or milestone representations [9], Durusau and O'Donnell's BUVH and JITTs [3], or simply a set of XML files (one file per encoding hierarchy). 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 [10], 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 [5].

- *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 [6], such as the **overlapping axis**. We show an efficient implementation of the Extended XPath and how it can be used to answer meaningful 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 appropriate 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[4].

- *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.

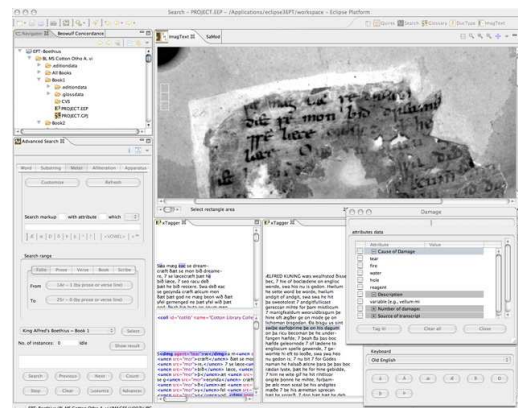


Figure 4: The EPT overview

The software we demonstrate 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, 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.
- [5] I. E. Iacob, A. Dekhtyar, and K. Kaneko. Parsing Concurrent XML. In *Proceedings WIDM*, pages 23–30, November 2004.
- [6] 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>.
- [7] 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.
- [8] 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. (Editors: N. Ide and S. Hockey).
- [9] 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.
- [10] C. M. Sperberg-McQueen and C. Huitfeldt. GODDAG: A Data Structure for Overlapping Hierarchies. In *DDEP/PODDP*, pages 139–160, Sept. 2000.
- [11] A. Witt. Meaning and interpretation of concurrent markup. In *Proc., Joint Conference of the ALLC and ACH*, pages 145–147, 2002.