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

We present the suite of document-centric, concurrent XML processing tools developed as part of the interdisciplinary ARCHway project. The goal of the ARCHway project is to develop software for production and presentation of Image-Based Electronic Editions (IBEEs) of historic documents. XML encodings of the document features form concurrent hierarchies, and markup in these hierarchies often overlaps, making traditional XML tools not immediately applicable. In this project we have developed the means for representing concurrent XML markup hierarchies and querying their content.

1. INTRODUCTION

In recent years XML has been actively used by the humanities scholars in preparation of Image-Based Electronic Editions (IBEEs) of historic documents. Such editions integrate the known text of the document, the high-resolution images of the, typically hard-to-access, original document and the editorial comments in one software package, accessible to the researchers from their desktop computers. XML is used as the means of encoding diverse features of the documents, from physical layout of the text on the document folios, to text structure, condition, restorations, paleography and editorial emendations.

The work of the interdisciplinary NSF-ITR-sponsored ARCHway project at the University of Kentucky concentrates on building software tools (and solving underlying computer science problems) for building and presenting IBEEs. IBEEs are built by human editors – humanities scholars with the knowledge of the document and the expertise to precisely describe its features. The editor's software tools produced within the ARCHway project facilitate the work of the editor on encoding the edition text based on the document images. The second suite of tools is devoted to presenting the prepared IBEEs to the users – other humanities scholars with the interest in studying the document described.

The editor's tools developed in the ARCHway project use XML to encode the features of the document. Key functionality of the presentation suite is the ability to search accumulated XML markup and deliver search results in efficient and convenient manner. In achieving this, the XML search (querying) tools of the ARCHway project have to deal with one significant obstacle: the XML markup describing the underlying document is multihierarchical, and features from different hierarchies often overlap in their scope. Because such markup is dictated by the needs of the editors, it falls on the shoulders of computer scientists to come up with appropriate solutions.

The problem of management of multihierarchical (also called concurrent) document-centric XML markup has been known to humanities scholars for years, first mentioned in the context of SGML [11]. Proposed approaches to dealing with concurrent markup [12, 6, 4] concerned representation of concurrent hierarchies but did not address the management/querying problems from the computer science point of view. In [2, 8, 3, 9] we have described how concurrent XML hierarchies can be represented in main-memory data structures, automatically managed and queried using an enhanced version of XPath.

In this demo, we present the software built on the basis of [2, 8, 3, 9] for the ARCHway project. Part of both editorial and presentation tools in ARCHway, implemented in Java Eclipse platform, the software manages the wealth of editorial markup created the human editor, archives it for further use and provides efficient query facility to the edition markup. The document-centric XML editor, which lies at the core of the ARCHway editorial tools uses a Generalized Ordered-Descendant Directed Acyclic Graph (GODDAG) [13] representation for concurrent XML (in [8] we report on the implementation of the GODDAG parser and GODDAG API used in the software). On top of the GODDAG API we have defined and implemented the XPath processor [9] enhanced with the functionality for detecting relationship between elements in different hierarchies, and finding overlapping markup. To our knowledge, this is the first software package that can (i) parse concurrent XML documents, (ii) store concurrent XML documents in a single memory-

*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 Baltimore, Maryland USA

resident data strucure, (iii) provide DOM-styles API access to this data structure and, finally (iv) efficiently process (enhanced) XPath query over the concurrent XML markup.

This paper is organized as follows. In the rest of the introduction we describe in more detail the propreties of the document-centric XML markup and our motivation for the project. In Section 2 we give a formal description of concurrent markup and our approach to dealing with it. We give the overview of the main features of our software in Section 3.

1.1 Document-centric XML markup

XML owns its growing popularity as the information representation medium 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. More and more information gets encoded in XML every year.

Two major types of XML result from using XML as the representation medium for semistructured data and as the means for encoding meta-information about documents: *data-centric* and *document-centric*. Data-centric XML typically represents a collection of semistructured data objects, where different levels of the XML element hierarchy represent objects and their features, while the content of XML elements represents the values of the features. Often the order in which elements occur in the data-centric XML documents at the same level is not relevant, thus the following two XML fragments

```
<name>                          <name>
   <first>John</first>             <last>Public</last>
   <middle>Q.</middle>             <first>John</first>
   <last>Public</last>             <middle>Q</middle>
</name>                         </name>
```

are frequently considered to be equivalent. Document-centric XML documents typically contain XML elements with mixed content, that describe properties, features, and/or meta-information about a pre-existing textual document. Note that the union of content of all XML elements forms the text being encoded. Thus, the order in which elements and their content exist in a document-centric XML document is unique. The following two fragments are distinct document-centric XML documents, despite the fact that the content of each XML element is the same:

```
<sentence>The <adj>quick</adj> <adj>brown</adj>
     <n>fox</n> jumps over a <adj>lazy</adj>
     <n>dog</n>. </sentence>

<sentence>The <adj>brown</adj> <adj>lazy</adj>
     <n>dog</n> jumps over a <adj>quick</adj>
     <n>fox</n>. </sentence>
```

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.

1.2 Motivation

The problem of overlapping markup 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: 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.* For instance, let us combine the two sentences in the example above in three physical lines and markup the lines (we remove the other markup for the clarity of this example):

EXAMPLE 1.

```
<sentence><line no="1">The quick brown fox
   jumps over a</line>
<line no="2">lazy dog.</sentence>
   <sentence>The brown</line>
<line no="3">lazy dog jumps over a
   quick fox.</sentence></line>
```

As we can easily see in the example, there are two conflicts between `<sentence>` and `<line>` markups: in the second and third `<line>` respectively. If the conflict in the third `<line>` can be easily removed by switching tags places, for removing the conflict in the second `<line>` there is no straightforward solution. We must observe that the concurrent markup makes the document non well-formed, hence no traditional XML software can be use to process the document (in fact, despite using XML-like tags, the fragment in the example above is not an XML document [1]). Humanities researchers [12, 6, 4] proposed various way of storing concurrent markup in a single XML document. Unfortunately, until recently, there have been no support for management of concurrent XML in software appliactions.

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.

2. 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 $CMH = \langle \rho, \{T_1, T_2, ..., T_k\} \rangle$ 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

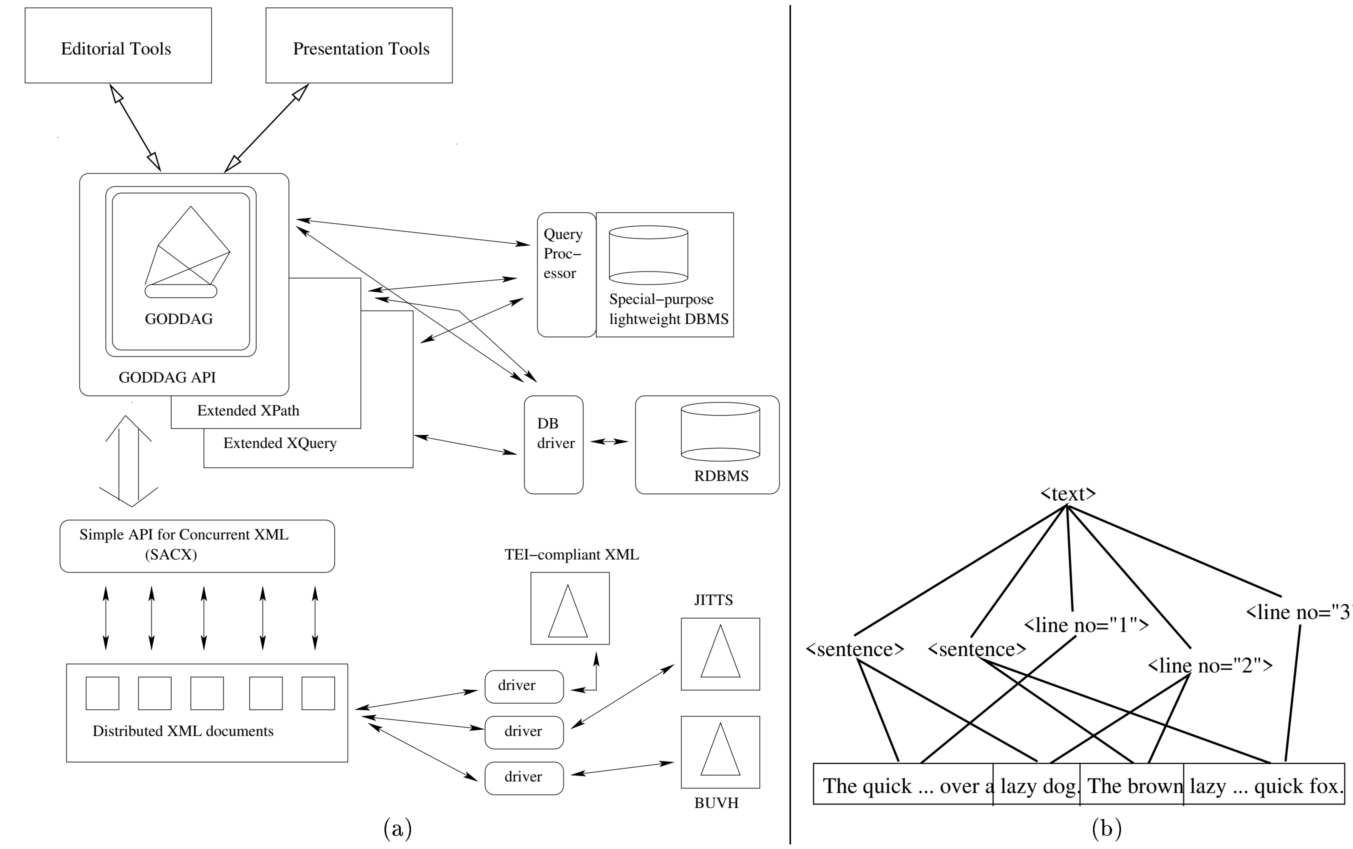

Figure 1: The framework for management of concurrent markup (a) and the GODDAG data structure for the text in Example 1 (b)

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* $root(D) = root(d_1)$*.*

Each markup structure is represented as a DOM tree, 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 [13]). For instance, the GODDAG of the document with overlapping structure in Example 1 is given in Figure 1.(b). 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 1(a)) 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.

3. DEMONSTRATION OVERVIEW

The following problems are addressed in our framework:

- *Data model for concurrent document-centric XML.* We use the GODDAG data structure [13] to represent concurrent XML documents, in the same way DOM trees are used to represent XML data. We design [8] mechanisms for parsing textual concurrent XML data into GODDAG, and we describe and implement the DOM-style API for the GODDAG representation. Our demo includes parsing a GODDAG from a set of XML files with different encodings. We demo also parsing a GODDAG from a single XML file where conflicting markup were represented as milestones (empty tags).

- *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, and extend it with features that are specific to processing of concurrent XML, such as the `overlapping` axis. For instance, we can find all line numbers for lines overlapping the first sentence in text using an extended XPath Query:

```
//sentence[position()=1]/overlapping::line/@no
```

We developed and implemented efficient algorithms for processing extended XPath queries over GODDAG structures [9].

- *Pre-Validation of document-centric XML documents.* In text encoding applications, document-centric XML is often created via prolonged manual editing of the underlying textual content. In such cases, the document might not become valid until very late in the editing process. At the same time, human editors introducing markup into the document need to know whether or not, the current state of the document allows for a valid extension. In [7] we introduce the notion of *potentially valid* XML documents (document that are to become valid if more markup is introduced at appropriate positions) and give a preliminary study of the problem of potential validity.

- *Manipulation of the documents in a concurrent XML hierarchy.* One of the advantages of the concurrent XML formalism we propose in this work is its flexibility: concurrent XML can, in theory, be imported into/exported from our framework from/in a wide range of alternative representations ([2]). We demo the filtering feature for partially viewing the document encodings or to export a subset of document encodings.

- *Special-purpose, lightweight DBMS for Concurrent XML.* It is often the case that the software required for some projects (like for instance, the Electronic Editions of Old English manuscripts) needs support for efficient management of large quantities of data, but, also must be independent of large commercial DBMS. We have implemented specialized data and index structures for storage of concurrent XML on disk. The necessary query support is also provided (extended XPath).

The framework we demo is completely implemented in Java on Eclipse platform, as an Edition Production Technology toolkit for creating image-based electronic editions of manuscripts.

4. CONCLUSIONS

The problem of concurrent markup hierarchies often occurs in XML text encoding applications in humanities. Previous solutions to the problem rely on manual maintenance of the markup and address only the problem of representing overlapping features in XML. As a consequence, processing documents using traditional XML tools is of little practical use when overlapping features are

In this work we propose a framework for management of concurrent XML hierarchies from a computer science perspective. We propose a model, data structures, and algorithms so that the computer would support most of the burden of managing concurrent XML hierarchies.

5. REFERENCES

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. C. (Eds.). Extensible Markup Language (XML) 1.1. http://www.w3.org/TR/2004/REC-xml11-20040204, Feb 2004. W3C Recommendation 04 February 2004.

[2] A. Dekhtyar and I. E. Iacob. A Framework for Management of Concurrent XML Markup. *Data and Knowledge Engineering*, 2004. accepted.

[3] A. Dekhtyar, I. E. Iacob, J. W. Jaromczyk, K. Kiernan, N. Moore, and D. C. Porter. Database support for image-based electronic editions. In *Procedings, 10th International Workshop on Multimedia Information Systems (MIS 2004), College Park, MD*, August 25–27 2004.

[4] P. Durusau and M. O'Donnel. Declaring Trees: The Future of the Evolution of Markup? In *Proc. Conference on Extreme Markup Languages*, 2002.

[5] P. Durusau and M. O'Donnell. Declaring trees: The future of the evolution of markup? In *Proc. Conference on Extreme Markup Languages*, 2002.

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

[7] 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.

[8] I. E. Iacob, A. Dekhtyar, and K. Kaneko. Parsing Concurrent XML. In *Proceedings, 6th ACM International Workshop on Web Information and Data Management (WIDM 2004), Washington, DC.*, November 2004.

[9] 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.

[10] A. Renear, E. Mylonas, and D. Durand. Refining our notion of what text really is: The prob lem of overlapping hierarchies. *Research in Humanities Computing*, 1993. (Editors: N. Ide and S. Hockey).

[11] 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.).

[12] 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.

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

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