

XML Technologies for the Digital Library

Timothy W. Cole, William H. Mischo, Thomas G. Habing, and Robert Ferrer

Grainger Engineering Library Information Center

University of Illinois at Urbana-Champaign

thabing@uiuc.edu

http://dli.grainger.uiuc.edu/

ABSTRACT: Digital formats used to represent text objects continue to evolve. Newer markup formats such as Extensible Markup Language (XML) support encoding of texts as “ordered hierarchies of content objects” and are a sophisticated and powerful way to represent text digitally.¹ XML has made the best features of Standard Generalized Markup Language (SGML) more accessible to Web authors and publishers, but the specification doesn't explicitly deal with presentation, nor does it address transformation of text. These issues must be addressed using stylesheets – e.g., Cascading Stylesheets (CSS) and Extensible Stylesheet Language (XSL) stylesheets. This paper describes how XML, XSL, and CSS can be used together in a digital library application. We focus on techniques used to transform SGML into well-formed XML, to render mathematics using CSS, and to transform XML using XSL.

KEYWORDS: Electronic Publishing, DL design methodology, DL testbeds, XML, XSL, CSS

THE TESTBED: A digital library testbed containing SGML-formatted journal articles was created as part of a Digital Library Initiative I project at the University of Illinois at Urbana-Champaign (UIUC).² As part of the ongoing Corporation for National Research Initiative's D-Lib Test Suite program this testbed is continuing and has been transformed into XML. Four professional society publishers contribute Testbed content: the American Institute of Physics (AIP); the American Physical Society (APS); the American Society of Civil Engineers (ASCE); and the Institution of Electrical Engineers (IEE). The Testbed contains some 55,000 articles from 44 journals. A Web-based search and retrieval interface, DeLiver (Desktop Link to Virtual Engineering Resources) has been implemented as part of the Testbed. Contributing publishers and other collaborating partners provide additional monetary and in-kind support.

TRANSFORMING SGML INTO WELL-FORMED XML: We continue to receive source materials from our publisher partners in SGML. They have much invested in SGML authoring and pub-

lishing tools, and equivalent XML tools lag in terms of functionality. Fortunately, the SGML feature set used by our publishers is relatively congruent with the feature set available in XML.

In transforming valid SGML into well-formed XML (i.e., XML not requiring a Document Type Definition) we begin by maximizing the portions of each document instance that are simultaneously SGML and XML compliant syntactically. For instance, we transform all EMPTY content model instances since XML and SGML applications differ in how such models are implemented. Thus we use only the third structure shown below –

<EmptyTag>	(SGML)
<EmptyTag />	(XML)
<EmptyTag></EmptyTag>	(SGML & XML)

We also add a closing “?” character to all processing instructions (required in XML and acceptable in SGML). We enclose CDATA content using the appropriate XML markup (<![CDATA[...]]>). Finally, we transform entity references. While named entities can be used in validated XML, they are not supported in well-formed XML. Fortunately, Unicode numeric entities are allowed. To preserve original markup information, SGML named character entities are converted into placeholder tags containing no content but having attributes that specify the SGML named entity, the equivalent Unicode entity, and the paired font face and ASCII code of the desired glyph. Tags having attributes as suggested by the proposed XLink standard are used to replace external object entities (e.g., graphics).

At this point the document instances may no longer be valid SGML, but assuredly are well-formed XML. They can easily be transformed into valid SGML (e.g., renderable by a viewer such as Interleaf's Panorama), into HTML / XHTML, or into XML suitable for rendering by an XML-enabled browser such as Microsoft's Internet Explorer version 5 (IE5).

WELL-FORMED VS. VALID: While creation of XML DTDs for documents in the Testbed would simplify entity handling, the differences between XML and SGML DTD syntax (e.g., XML DTDs do not

permit inclusions, exclusions, or ordered mixed content models) make translation of publisher SGML DTDs difficult. Also, because of XML DTD limits, any validation benefit would be minimal. Given the likelihood that XML Schemas will supplant XML DTDs, the decision was made to defer translation of SGML DTDs for the time being.

RENDERING MATHEMATICS: One of the major deterrents to publishing scientific literature on the web has been the difficulty of rendering mathematics natively in web browsers (other than as embedded bit-maps). Although cross-browser and cross-version compatibility continue to be major concerns, wider and more complete implementations of JavaScript, Unicode, downloadable fonts, and standards such as CSS version 2 are helping to lessen the difficulties. These technologies support the level of formatting and positioning needed, although implementation remains complex. Consider:

$$\frac{1}{3} T_F$$

The as-provided mark-up for this clause is:

```
<formula><f><fr>
<nu>1</nu>
<de>3</de></fr>
T<inf>F</inf></f></formula>
```

XML to XML transformation algorithms (optimized for more complex equations) transform this to the following for rendering in IE5. Note the use of HTML namespace (for tags with inherent functionality).

```
<formula><html:nobr><f><fr>
&#160;<html:span id='dli10' class='fr'>
<nu><norm>1</norm></nu>
<de><hidden_sup>|</hidden_sup><norm>3</norm>
<hidden_inf>|</hidden_inf></de>
</html:span>&#160;
<html:script language='JavaScript'>SetWidth("dli10");
</html:script></fr>
T<inf>F</inf></f></html:nobr></formula>
```

To insure proper rendering considerable presentational markup is added. An HTML no break tag is added to force the browser to render the entire mathematical clause on one line. Non-breaking spaces () and hidden super and subscripted vertical bars (|) are added to ensure proper horizontal and vertical placements. An HTML span with a unique id attribute and a class attribute equal to 'fr' is added within the <fr> element to allow the fraction width to be set by a JavaScript function (SetWidth). Numerals are enclosed within a <norm> tag so they can be rendered in a non-italic style. An associated

stylesheet (CSS) then provides additional rendering instructions, adjusting font sizes and styles and displaying <nu> and <de> elements as stacked blocks within the width-constrained <fr> element. The bottom border of the <nu> becomes the horizontal line of the fraction.

While the above approach is powerful, works well for other constructs, and can create HTML that renders well, there are limitations. The quality of the result, while understandable, does not approach the quality of a dedicated mathematics typesetting system, printed mathematics, or specialized plug-in applications. Font issues remain. Results are not as good for browsers that support fewer features of CSS ver. 2 (e.g., Netscape 4.x browsers). Additional standardization of markup (e.g., widespread implementation of MathML), more math-specific public domain fonts, and further browser enhancements are still needed.

USING XSL: XSL facilitates XML to XML transformations (e.g., as described above), XML to HTML transformations, and selective extraction of data from XML files. In combination with test and pattern matching rules, the basic XSL elements (e.g., <apply-templates>, <attribute>, <choose>, <copy>, <foreach>, <if>, <template>, <value-of>) provide most functionality required. However, in some cases it is desirable to do direct manipulation of XML element content and/or access external objects (e.g., databases) using XSL. The current W3C specifications do not yet address these matters fully. Microsoft has unilaterally provided methods for invoking scripts and COM objects from within XSL, and we're exploiting these methods pending release of newer standards.

We use XSL stylesheets to assist in creating metadata records, to transform those metadata records to HTML for display by standard Web browsers, and to transform our base XML files as necessary to support rendering of Testbed articles. XSL templates can be applied real-time at the client (by XSL-aware browsers such as IE5) or at the server (as content is requested). Though implementations of XSL are still limited and immature, we're already able to make excellent use of the partial implementations available.

¹ Steven J. DeRose, et al, "What is Text, Really?" *The Journal of Computer Documentation (ACM SIGDOC)* 21, no. 3 (August 1997): 1 -- 25

² Bruce Schatz, et al, "Federated Search of Scientific Literature: A Retrospective on the Illinois Digital Library Project," *IEEE Computer* 32, no. 2 (February 1999): 51-60.