Preserving and Publishing Digital Content Using XML Workflows

Jonathan McGlone

University of Michigan Libraries, Michigan Publishing

3

IN THIS CHAPTER

Theme

Production streamlining

Highlighted Services

Online journal & monographs hosting and production

Software/Platforms Utilized

Drupal, DLXS, Adobe InDesign, Calibre

Resources

Example XML workflows

In digital publishing, encoding documents in XML can produce several advantages for libraries that have invested in hosting and publishing services or plan to in the future. XML workflows enable publishers to output content quickly and easily in several electronic formats (EPUB, HTML, PDF); repurpose content into other channels (catalogs, websites, databases, printers); automate processes; scale their services and publications; and preserve the digital content for the future.

Michigan Publishing (formerly known as MPublishing), the primary publishing unit of the University of Michigan and a part of its University Library, began encoding born-digital documents in SGML—and later XML—to publish journals and monographs in the late 1990s. Over time, Michigan

Publishing has established its own semi-automated XML workflow to achieve these ends in its work with a growing list of publishing partners. Today Michigan Publishing provides web-hosting and conversion services for over 20 active open access and subscription-based academic journals (http://www.publishing.umich.edu/publications/journals), the open access Digital Culture Book Series (http://www.digitalculture.org), and the open access imprint Open Humanities Press (http://openhumanitiespress.org/book-titles.html), among others.

The following chapter will introduce the reader to a few flavors of XML encoding used at Michigan Publishing such as JATS XML, Scribe, and TEI, as well as present a generalized workflow for creating and publishing encoded XML documents for monographs and journal articles that begins with a Word document, outputs an XML encoded document, and presents the article or monograph to readers as HTML and PDF, yet is also available for repurposing into other channels.

Before You Start

Before you start a digital publishing project, determining how the content will be used by readers and the project's preservation requirements are an important first step. Does the content need to be searchable? Will it offer content in multiple formats—PDF, mobi, EPUB2, readable on the Web as HTML? Is there interest in re-purposing specific pieces of the content in other channels such as a library vendor's database? Will you be registering DOIs (Digital Object Identifiers) for the content? Do you need to ensure that your content can be accessed and migrated to new formats five to ten years from now? Twenty years from now?

If your answer is yes to any of these questions, you'll want to strongly consider encoding your documents with a markup language such as XML. If you simply need to present images of the original pages (referred to as page images), then scanning, conducting OCR (Optical Character Recognition), and presenting content on the Web as downloadable PDFs with searchable OCR text may be enough. Because the upfront costs to establishing an XML workflow can be quite considerable and not every digital publishing project requires XML encoded documents, defining the scope and aims of the project as early as possible can help minimize any unneeded efforts, and help to focus your publishing project's production processes (Gross, 2003).

Benefits to Using XML

XML works best with text-based content that has structure. In the academic publishing world, a majority of content published is text based. What's more, traditional journal articles and scholarly monographs use a fairly predictable structure. This is especially true for journal articles.

Since the 1990s, academic publishers have been taking advantage of this predictable structure by implementing XML in production workflows. With XML, publishers can validate content and ensure it is in the correct order; for example, running tests to check that footnotes are at the end of a document or that an abstract appears before the first paragraph. Validation can also ensure there is consistency across multiple XML documents, making sure that headings or footnotes have been encoded the same way. When working with tight deadlines and several projects at once, these types of validation can save a lot of time.

Structure can also be relied upon to automate processes. When converting XML to other formats such as HTML, the structure can be used to programmatically apply character styles to section headings or chapter titles. Or when registering DOIs for journal articles, article metadata can be re-used to automate an XML submission to the DOI registrar, CrossRef.

From a preservation and access perspective there are several benefits to encoding your documents with XML. As an open file format, reading and exchanging your XML data can take place regardless of operating system, platform, or software. Because it is open, chances for the loss of data when sharing your XML content with library vendors or other institutions is greatly reduced. In addition, the open nature of the file format also ensures future access to your XML documents as long as there are programs that can read and write Unicode text.

What's more, because XML aims to keep content separate from the design elements (e.g., line spacings, margins, font types), your content can remain accessible as media and devices evolve and change. For example, if a new tablet reader format emerges, you won't risk the chance of losing any content when migrating to these new formats; you simply need to repurpose your XML to the new format's syntax.

In order for your XML content to be machine readable and interoperable, you'll want to select a Document Type Definition (DTD) that imposes rules upon how you mark up your content.

Most importantly, encoding documents in XML allows content to be repurposed into other formats using XSL Transformations (XSLT) or other programming languages. Using XSLT, you can move your XML into different XML Document Type Definitions (DTDs), such as InDesign's, for typesetting a PDF, HTML for Web reading, or e-book file formats such as mobi or EPUB. The ability to transform structured data into another format gets at the core of the convenience of an XML workflow in digital publishing.

A Few Flavors of XML Encoding Relevant to Academic Publishing

In order for your XML content to be machine readable and interoperable, you'll want to select a Document Type Definition (DTD) that imposes rules upon how you mark up your content. A DTD is a set of tag names and element attributes that are agreed upon so use and application of tags is the same across documents. While there are many DTDs to select from, the following section reviews those used in Michigan Publishing's journal and monograph workflows.

TEI and DLXS Text Class | http://www.tei-c.org

All of Michigan Publishing's journal content and some of its monographs are available online through a hosted installation of DLXS (Digital Library eXtension Service), digital library collection software developed at the University of Michigan. DLXS provides a storage mechanism for our XML encoded files and indexing and searching tools, as well as a set of customizable XSLT stylesheets to transform content from XML to HTML rendered in readers' web browsers.

Although DLXS has its own Text Class DTD consisting of basic requirements in order for the software to index, search, and transform the XML, we use our own in-house version of TEI to provide higher-level encoding of our documents (DLXS, 2009). In the end, the resulting XML document is TEI wrapped in the Text Class DTD. TEI best suits the needs of Michigan Publishing because of its flexibility and modularity, allowing for customizations related to our digital collection platform.

From the beginning TEI was designed by the humanities research community to be able to support as many kinds of materials as possible, striving to "be applied to any natural language, literary genre, text type without restriction on form or content" (Burnard & Bauman, 2013). TEI was originally conceived to describe print documents in as much detail as possible, representing in electronic form text that already exists in traditional media. Therefore, it is suited well for academic publishing materials, especially dictionaries and running text materials like monographs or journals. In addition to the support of a very active and knowledgeable community of practitioners, the TEI community has developed several conversion tools—to TEI XML and from TEI XML to other formats—that are designed for users to adapt and customize (see http://www.tei-c.org/Tools/Stylesheets/).

Journal Article Tag Suite (JATS) | http://jats.nlm.nih.gov/

The Journal Article Tag Suite (JATS) represents the emergence of a distinct set of XML elements and attributes aimed at academic journals. As a newly created NISO standard, it provides a common XML format for publishers and archives to exchange and preserve journal content. Like the majority of XML DTDS, JATS does not attempt to preserve the journal form or content style (NISO, 2013).

Prior to standardization, JATS was used as the National Library of Medicine DTD. Developed at the National Center for Biotechnology Information (NCBI) and the National Library of Medicine's PubMed Central, it began as a set for archiving life science journals and later expanded as needs grew. It is now used by journals worldwide, especially journal archives at PubMed Central, Portico, and HighWire Press, as well as the Library of Congress and British Library (Beck, 2011).

As Michigan Publishing evolves away from publishing digital content using DLXS and the Text Class/TEI DTD, JATS will become our future journals DTD. As a replacement to DLXS, Michigan Publishing is currently developing web-based tools to allow publishers to easily transform content from Microsoft Word to JATS XML and later deposit and publish this content in HathiTrust (http://www.lib.umich.edu/mpach).

Scribe Markup Language (ScML)

Unlike the other DTDs described above, Scribe Markup Language (ScML) is a proprietary XML format designed for moving documents from a digital to print environment. Specifically developed for the publishing industry, subscribing publishers/institutions have access to the ScML documentation and dictionary along with Scribe's document workflow tools. Scribe's pre-developed set of workflow and conversion tools are extremely useful for publishers looking to begin or transition into an XML workflow. More recently, Scribe has begun developing tools to convert content to e-book formats such as EPUB2.

Because it is designed specifically for publishers, the ScML DTD includes additional elements that indicate elements important to typesetters and the typesetting process, such as first paragraphs, paragraphs after heads, or those that continue onto pages (Scribe, 2013). Using a set of proprietary macros and plugins for Microsoft Word, styles are applied to the Word document manuscript to give the document structure (headings, first paragraph, etc.), which a programmed script interprets to transform the document to the corresponding ScML XML.

Michigan Publishing has adopted ScML and its workflow tools for its print projects under the University of Michigan Press imprint. In addition, adoption of this workflow has allowed Michigan Publishing to develop and offer "rapid typesetting" services for journal publishing partners—producing typeset PDFs for additional fees. Increasingly, with the addition of tools that assist in the conversion to EPUB2 and mobi, ScML will likely become a "pivot" format to produce a version for multiple platforms that also converts to JATS XML or DLXS Text Class/TEI as a preservation format.

Engineering an XML Workflow

There are two main approaches to XML workflows for digital publishing: XML-In and XML-Out. XML-In workflows involve the creation of the XML files at an early stage in the production process, such as authoring in XML (highly unlikely), having a copy editor prepare a word document for XML conversion, or prior to the design and typesetting process (Bullock, 2012). XML-Out workflows maintain a standard production processes—writing, editing, proofing, and typesetting—and XML is created in a back-conversion process (Strange, 2003, p. 158). Depending on the individual project or your relationship with the content provider, you may use one or the other.

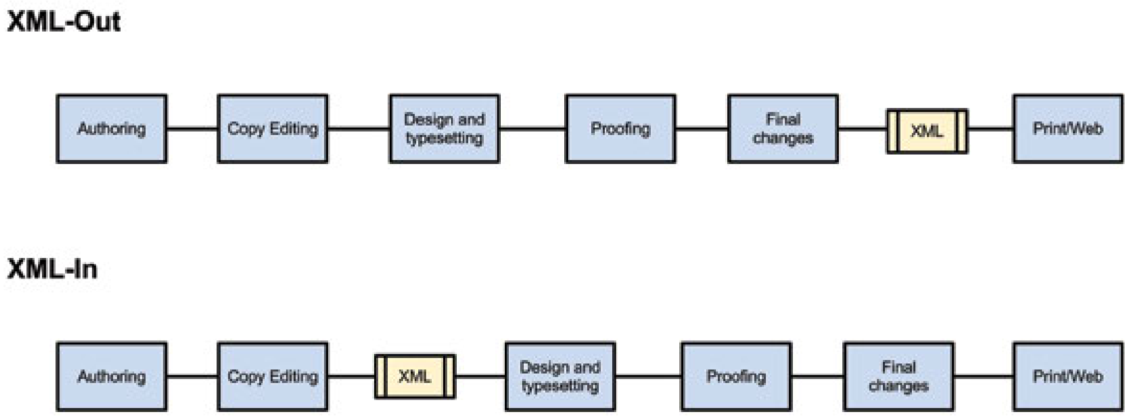

Figure 1. XML-In vs. XML-Out.

The sooner content can be converted to XML, the quicker and more efficient your workflow will become. For example, when generated early in the production process, XML can be converted to HTML for online review and proofing by editors and authors before it is typeset. The XML file becomes the source or master file for both your digital and print output. In addition, with an early XML workflow, the typesetting process can be automated using templates in programs like InDesign when importing XML content, a process that is especially useful for journals because of their consistent formatting across volumes and issues.

The digital publishing activities of Michigan Publishing have relied primarily on XML-Out because of our tendency to provide only conversion and hosting services for journals and digital monographs, similar to many libraries providing publishing services. But as demand for typesetting services, EPUB versions, and the need to minimize costs increase, adopting an XML-In workflow where copy editors or conversion assistants structure content in a Microsoft Word document by applying paragraph and character styles while reviewing content is likely to take place.

Basic XML-In Conversion Workflow

In a basic XML-In workflow (Figure 2), content is prepared by the author in familiar authoring tools such as Microsoft Word or Open Office, converted to XML, then transformed to produce PDF, HTML, and EPUB versions of the content.

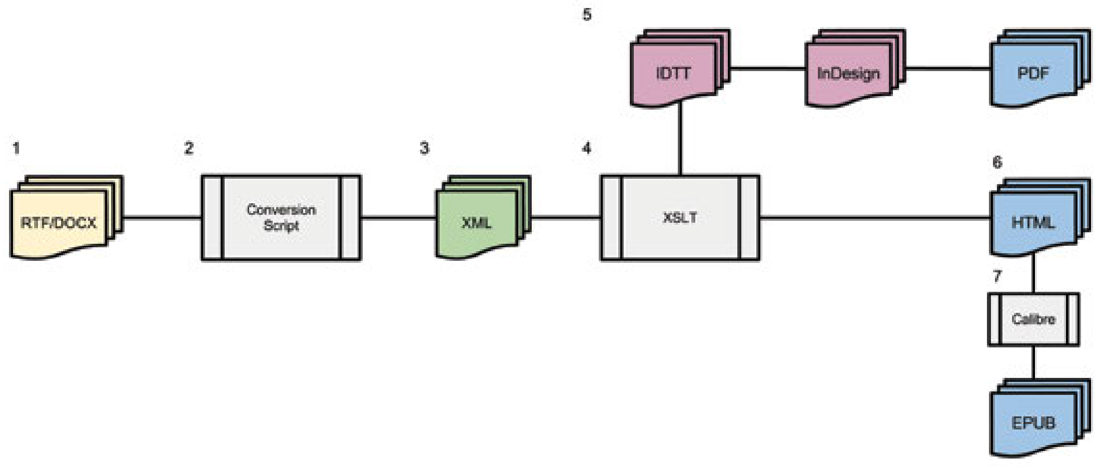

Figure 2. Basic XML-In Workflow. XML is generated after copyediting and just before design and typesetting. The HTML version produced here can be used as the proofing copy.

When receiving the content, customized paragraph and character styles can be applied to the RTF or DOCX file to identify elements such as the author's name, title of the chapter or article, footnotes, paragraphs, blockquotes, and section headings. These styles give a basic structure to the document, which aides a conversion script written in a programming language such as Perl or Python to identify and convert elements to their corresponding XML elements; e.g., paragraphs to <p> tags or blockquotes to <p type=''blockquote''>. Michigan Publishing uses the proprietary software R2Net (http://www.logictran.net/products/r2net.html) for command line RTF to XML conversion, which offers support for conversion to customized document types, which is essential in conversion to DLXS Text Class DTD. A free and open source alternative for RTF conversion exists in Paul Tremblay's rtf2xml project (http://rtf2xml.sourceforge.net/). In addition, the TEI Consortium has produced a XSL stylesheet to aide in the transformation of DOCX to TEI XML (https://code.google.com/p/oxygen-tei/source/browse/trunk/oxygen-tei/frameworks/tei/xml/tei/stylesheet/docx/from/docxtotei.xsl?r=9).

Once the content is identified in XML, further XSL transformations (XSLT) can be applied to convert the content to other markup languages, such as InDesign Tagged Text (IDTT) for importing into InDesign (see http://help.adobe.com/en_US/indesign/cs/taggedtext/indesign_cs5_taggedtext.pdf), or to any flavor of HTML for display on the Web. When content is in a program such as InDesign, content can be professionally typeset and a print-ready PDF can be generated. Because EPUB is a packaged version of HTML, once your content is in HTML format, converting to EPUB or one of a plethora of other e-book formats using software such as Calibre (http://calibre-ebook.com/) is ideal, as it can help preserve styling and formatting applied to the HTML. Again, the TEI community has developed a set of XSL stylesheets to aid in the transformation of TEI XML to other formats such as HTML, HTML5, EPUB, and EPUB3.

Journals (XML-Out Workflow)

Traditionally, Michigan Publishing's journals workflow has worked as a standard XML-Out workflow, primarily determined by our relationships with publishing partners. For example, journal editors conduct their peer review, copy editing, and in most cases, typesetting, independently. Once content has been finalized, it is submitted to Michigan Publishing based on pre-defined specifications for submitting content in the form of a checklist to ensure all figures, tables, images, and manuscripts are present and correctly named.

When providing hosting and conversion services, you need to decide how flexible you will be in what formats you will accept. If you have a loose policy on deliverable formats, be prepared to receive all different types of formats and content from partners. In the past, not wanting to preclude publishing based on a partner's established authoring and editing workflow, Michigan Publishing has allowed publishing partners to deliver manuscripts in the form of Word documents; Rich Text Files; PDFs; InDesign, Quark, and TIFF files; and bound volumes and paper documents, thus forcing the development of several different and diverging conversion workflows. As we are now operating at a larger scale, our preferred delivery formats for content bound for the Web are Microsoft Word documents or PDFs.

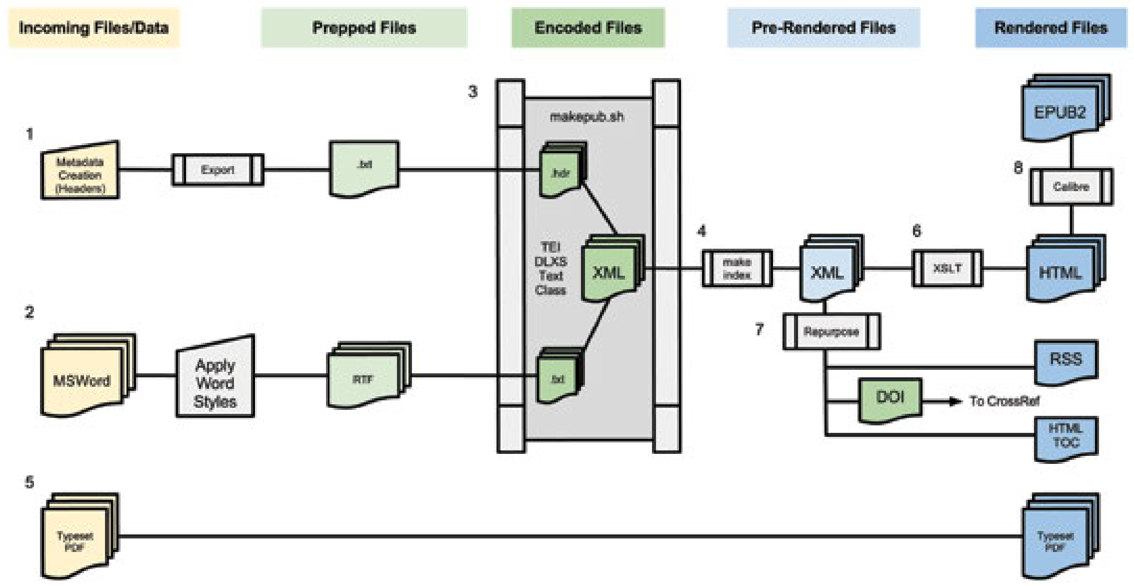

Figure 3. Michigan Publishing Journals XML Workflow.

Figure 3 demonstrates a typical journal workflow, where Word documents and a typeset PDF are delivered to Michigan Publishing to offer HTML, PDF and EPUB2 files available to readers. In Step One, metadata for each article is entered into a customized Drupal database, recording information such as the author(s), title, keywords, volume, issue, DOI (which will be registered upon publication), and other relevant article and issue information. Database metadata is exported to an XML encoded text file that serves as the header of article, later to be joined with content. Metadata is purposely stored separately from the content of each article to easily manage and apply global changes to journal information and to reduce the number of keystrokes by using calculated database fields.

Separately, a Word document is converted to an RTF file where it is given structure by applying custom Word styles to the elements of the article, as described in the previous section. Using a script developed at Michigan Publishing, the RTF file is converted to an XML encoded file, which is joined with a corresponding header and validated. Often, the conversion from RTF to XML is not a perfect process, requiring manual corrections before proceeding. Having a validation process in place ensures content is marked up uniformly and can be indexed and transformed to HTML correctly. After passing validation, it is indexed in the DLXS system. When a reader requests a specific article, a complex system of XSL stylesheet templates transforms the XML on the fly into HTML in the reader's browser. Using Calibre e-book management software, HTML content is converted to EPUB2 files, which are presented to readers for download in addition to or in place of a typeset PDF. In Step Seven, XML's custom Perl scripts are used to repurpose XML to generate a HTML table of contents for the current issue and RSS feeds, and automate the registration of DOIs with CrossRef. In addition to this type of repurposing, having our content in XML allows us to also submit content to bibliographic and full-text databases interested in indexing specific journals.

Monographs (Scribe Well-Formed Document Workflow, XML-In)

Michigan Publishing has published open access monographs on the web since 2001 using a variety of XML workflows, from back conversion of scanned PDFs to a similar process described above in Figure 3. Currently it is experimenting with a fully integrated XML-In workflow that utilizes Scribe's Well-Formed Document Workflow to aid typesetting, allow for in-house conversion to various e-book formats, and to present open access versions of texts to readers on the Web in HTML.

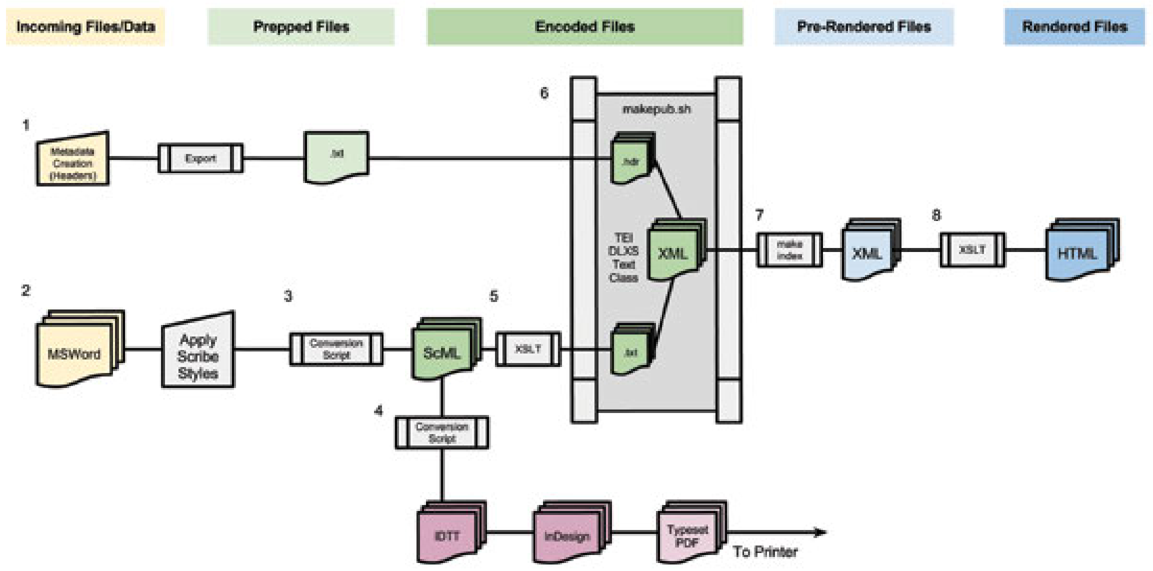

Figure 4. Michigan Publishing Desired XML-In Monographs Workflow.

Similarly to the workflow described in Figure 3, in Figure 4 basic metadata is keyed into a central database, which is later exported as marked-up TEI XML, merged with the body of the content, and used for the online version of the monograph. After receiving the draft manuscript in Microsoft Word DOCX format, copyeditors give structure to the document using Scribe-specific Word styles. This process is very similar to the process described in Figure 3, with the exception that the styles transform to ScML XML. After the copyediting process is completed and final, the DOCX is transformed to ScML and validated. Once in ScML, Scribe's conversion script generates an InDesign Tagged Text file for importing to InDesign, where images are inserted, the content is typeset, and the document is prepared for export to PDF and other formats for small-run printing or print on demand services. Although not utilized at this point, Scribe also provides tools to flow the typeset content from InDesign back to ScML for conversion to EPUB and other e-book formats. For web delivery, ScML is transformed to DLXS Text Class using XSL stylesheets, and run through the same process journals receive. It should be noted that with monograph publishing and production, the XML workflow should be flexible because monographs often have unique features and aspects that won't fit smoothly into your XML DTD or workflow.

Manual Intervention and Quality Control

While most of this process is automated, manual intervention and clean-up is always necessary during the conversion process. Often with RTF or DOCX to XML conversions for both journals and monographs, applied styles in Word do not get translated to the proper XML elements. This is where validation and error reporting post conversion can come in handy—if errors occur in the transformation, they will be identified and can be manually corrected before proceeding. It should also be stressed that a round of quality control or review of HTML is always necessary when converting content to ensure that XML elements are being transformed and rendered in the web browser correctly.

It should also be stressed that a round of quality control or review of HTML is always necessary when converting content to ensure that XML elements are being transformed and rendered in the web browser correctly.

The Future of XML Workflows

XML is well suited for publishing traditional scholarly content in multiple formats, repurposing into multiple streams, and preserving content for the long term. While these needs will continue to exist, new and emerging forms of scholarly communication will require new and different workflows. From another perspective, existing XML workflows should not constrain authors who want to experiment in new forms of publishing that might not fit into traditional production models.

For example, does it make sense to migrate content such as blog posts from HTML to XML when re-publishing them in electronic form? If a publisher is opting to take on diverse projects where content layout and format varies from manuscript to manuscript, does it make sense to impose a strict XML workflow on these projects (Daly, 2013)? What about projects that use HTML5 for video and user interaction? Beginning these projects with XML may not be a worthwhile effort because it can't represent video and coded interactivity. If EPUB3 allows for both JavaScript and HTML5 and does not utilize XHTML, what is the point of using XML at all? In short, as scholarly web publishing begins to rely on various types of content, interaction, and multimedia, XML and XML-based workflows may not always be the most efficient answer for non-traditional scholarly publishing.

References

Beck, J. (Summer 2011). NISO Z39.96 the Journal Article Tag Suite (JATS): What happened to the NLM DTDs?. *The Journal of Electronic Publishing*, 14(1). Retrieved from http://dx.doi.org/10.3998/3336451.0014.106

Bullock, A. (2012). *Book production*. London: Routledge.

Burnard, L., and Bauman, S. (Eds.). (2013). *TEI: P5 guidelines* (2.3.0 ed.). Charlottesville, Virginia: Text Encoding Initiative Consortium. Retrieved from http://www.tei-c.org/release/doc/tei-p5-doc/en/Guidelines.pdf

Daly, L. (2013, February 1). The unXMLing of digital books. *Safari books online: Publishing & technology.* Retrieved from http://techblog.safaribooksonline.com/2013/02/01/the-unxmling-of-digital-books/

DLXS (2009). Working With Text Class Markup. Retrieved from http://webservices.itcs.umich.edu/mediawiki/dlxs15/index.php/Working_with_Text_Class_Markup

Gross, M. (2003). Data capture & conversion. In W. E. Kasdorf (Ed.), *Columbia guide to digital publishing* (pp. 179–218). New York: Columbia University Press.

Kay, M. (2008). XSLT 2.0 and XPath 2.0 programmer's reference. National Information Standards Organization (2012, August 22). *NISO publishes Journal Article Tag Suite (JATS) Standard: Provides common XML format for exchanging journal content.* Retrieved from http://www.niso.org/news/pr/view?item_key=d92a2bc93b43db6831e68914e134c731d83cbdd1

Scribe. (2013). Scribe Well-Formed Document Workflow. Retrieved from http://scribenet.com/about/well-formed-document-workflow

Strange, J. (2003). Organizing, editing, & linking content. In W. E. Kasdorf (Ed.), *Columbia guide to digital publishing* (pp. 155–178). New York: Columbia University Press.