Handling Math in Real-World Workflows: Practical Lessons

Bob **Mathews**

Robert **Miner**

Abstract

The use of XML in scientific, technical, and medical (STM) publishing is growing, as more and more publishers seek to take advantage of the benefits of XML. One of the unique challenges of STM publishing is handling embedded mathematics. This paper will document some practical lessons Design Science has garnered from working with early adopters of MathML technology in STM publishing.

Table of Contents

1. Introduction

Scientific, technical, and medical (STM) publishers are increasingly seeking to take advantage of the benefits of XML. The presence of mathematical formulas is a chief characteristic of STM documents, and one of the unique challenges of STM publishing is how best to handle embedded mathematics. While MathML[MathML] has become firmly established as the standard way of representing mathematics in XML documents, a plethora of nuts-and-bolts questions remains about the best way to handle math in a real-world workflow.

2. Challenges Abound

The first challenge in setting up an STM workflow is deciding how MathML will be used. In setting up a DTD or schema, a publisher must decide how to incorporate MathML into the document level markup. Should you use a namespace prefix or not, and what implications will that have for configuring common MathML software tools? Should you be using MathML presentation or content markup?

2.1. DTDs and schemas in practice

Most tools for document-centric XML are still based on DTDs. There is a MathML schema[1], and a few projects have used it, mostly those using XML Spy[2] since it prefers schemas. Word, when used as an XML editor, is also based on schemas. However, most workflows we know of use DTDs. The most common way of adding MathML to a base doctype (such as DocBook) is to incorporate it using a namespace prefix (e.g., `<m:math>` or `<mml:math>`). That practice is fairly widely supported, and is the default for MathFlow[3] with Epic, XMetaL, and XPP. Drawbacks are that it's even more verbose, and it requires document level declarations in the markup, which makes the math less self-contained.

A minority of applications use a namespace attribute on the `<math>` element (e.g., `<math xmlns="http://www.w3.org…">`). That is the convention in the Mozilla/Firefox world. Since the namespace URL is long and error prone, the Mozilla people popularized the practice of defining it as an entity (e.g., `<math xmlns="&mathml;">`). The entity is typically defined in a local subset of the DTD in the document. It's a slick solution, but requires tools that can deal with definitions in the local subset, which not all can. In fact, namespace support itself is still sometimes a problem. The implementation in Microsoft Internet Explorer (IE) is non-standard for example, which affects MathPlayer, since it must use the IE mechanism.

A few workflows we have encountered simply quote all or part of the MathML DTD within an in-house DTD. A few others have extended MathML in unique ways (e.g., to permit foreign markup within MathML elements). There may be reasons to take these approaches, but most MathML tools can't handle such non-standard ways of using MathML, so the user should have a compelling reason to take either of these routes.

2.1.1. Presentation or content markup?

Most current STM workflows incorporating MathML use presentation markup. In general, the tools for presentation markup are better developed than those for content markup, and in most cases the visual presentation of the mathematics in print is the paramount concern. Within the last year or so, however, there has been an increase in interest in content MathML, mostly from publishers of elementary education content as well as e-learning systems that need to interpret the meaning of mathematics (e.g., for automated testing, or to analyze student work to provide targeted help). The main driver for elementary education content seems to be accessibility, of which more later.

[1] In this document, "schema" refers to a document defined in accordance with the XML Schema 1.0 [http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/].
[2] XML Spy is published by Altova: http://www.altova.com [http://www.altova.com/products_ide.html].
[3] MathFlow is published by Design Science, which also employs the authors. http://www.dessci.com [http://www.dessci.com/en/products/mathflow/].

3. Bringing documents into the workflow

Once you've settled on a document type, you need to bring documents into the workflow. In STM publishing, most documents begin as Word or LaTeX documents. What are the choices for converting the math into MathML? How does that integrate with conversion of the rest of the document? How well can I expect automated conversion to work, and what tools and techniques are available for cleaning up the results?

3.1. Conversion issues

There are many good tools for converting Word documents into XML, and a number of them support conversion of Equation Editor/MathType equations into MathML. All of them ultimately rely on MathType conversion facilities. MathType can convert all the equations in a document into a textual form (either MathML or TeX/LaTeX). Thus most Word-to-XML converters invoke MathType as a preprocessor to convert the equations to MathML, and then convert the resulting Word document in their usual way. Epic Interchange + MathFlow Exchange works this way. Another popular converter that uses MathType is Inera eXtyles[4].

The main problem with converting Word documents is not software but authors. Documents that have been carefully authored to be regular in their format tend to convert well. However, most Word documents are not carefully authored. The author often enters simple math expressions as a mix of symbols entered directly into Word, and Equation Editor or MathType equations for the more complicated bits. Similarly, the document text may look uniform, but is really a mix of many styles. Consequently, a number of production workflows have manuscripts cleaned up or re-keyed, and lock them down to an XML schema. A highly-skilled editorial staff then performs copy editing within Word, taking care with the underlying structure.

LaTeX conversion is harder. There are three or four main LaTeX-to-XML+MathML converters. TtM[5] is a fast, commercial converter, but its output is idiosyncratic and it isn't very configurable. TeX4ht[6] is an open source converter that is highly configurable. It works by redefining certain low-level LaTeX macros to insert extra information into the DVI output of the regular LaTeX compiler, and then post-processing the DVI into XML+MathML. The advantage is that it can process almost any LaTeX source file, but the down side is that reconstructing the logical structure from the DVI output is a little hit or miss. A new LaTeX converter called Hermes[7] is still in development, but seems promising. It is currently best run on Linux hardware. If authors use special macro packages, Hermes can output content MathML markup, which is basically unique among TeX converters. There are several converters, including WebEQ, that convert only the math subset of LaTeX and these are often useful in situations where math is being handled separately from document content in a workflow, or web application. For example, e-learning courseware frequently handles math separately.

In general, both Word and LaTeX conversions are difficult, time- and labor-intensive tasks, except in rare situations where the authoring is rigorously controlled. Consequently, virtually all of the production workflows with which we are familiar outsource the conversion of manuscripts. There are many conversion vendors with custom tools and in-house expertise, and in most cases the economics of conversion make this the best option for STM publishers.

3.2. Copy editing

Copy editing is usually the next phase in a typical workflow. In this phase, some of the most important challenges relate to skills and not software. Do your copy editors need to know MathML, or can you get away with relying on WYSIWYG

[4] eXtyles is published by Inera Inc. http://www.inera.com [http://www.inera.com/].

[5] TtM translates from Plain TeX and LaTeX into HTML including the equations in the form of embedded MathML. http://hutchinson.belmont.ma.us/tth/mml [http://hutchinson.belmont.ma.us/tth/mml/].

[6] TeX4ht is a highly configurable TeX-based authoring system for producing hypertext. It interacts with TeX-based applications through style files and postprocessors, leaving the processing of the source files to the native TeX compiler. Consequently, TeX4ht can handle the features of TeX-based systems in general, and of the LaTeX and AMS style files in particular. http://www.cse.ohio-state.edu/~gurari/TeX4ht/mn.html.

[7] Hermes is a semantic XML+MathML+Unicode e-publishing/self-archiving tool for LaTeX authored scientific articles. http://hermes.aei.mpg.de/workshop [http://hermes.aei.mpg.de/workshop/].

tools? How much and what kind of training should you do? What can you do to help prevent different people from marking up the same equation in differing ways that will only show up later during production?

The transition to XML is usually challenging for editorial staff. Editors often already possess highly-developed skills tied to particular software systems, and starting again with a completely new paradigm is difficult. Moreover, most older software systems for mathematical typesetting emphasize visual appearance almost exclusively, so moving to a more abstract XML way of thinking with less direct control over typesetting may seem a little awkward. The fact that even WYSIWYG editors typically show only an approximation of the final rendering (at least for some output media types) is frustrating for professionals used to kerning characters a fraction of a point to fine-tune mathematical typesetting. Consequently, experience indicates that training is important when moving to MathML-based workflows. Most organizations should expect 1-2 days of intensive training followed by several months of ramping up skills, and further followed by a refresher training course 1-2 years after initial training.

3.2.1. The importance of MathML experts

Like all XML languages, MathML is verbose and not intended for direct human editing, therefore WYSIWYG tools are essential for productivity. The majority of editorial staff likely will not need to delve deeply into the markup behind the tools. However, it is very important that organizations have one or two experts that can debug problems at the markup level. Most organizations moving to MathML are doing so in part to gain the benefits of more uniform and efficient processing of math in documents with less handwork. To achieve that, most workflows will require some sort of quality control on the MathML. Some of the things that organizations are doing (besides standard XML validation) are using customized macros, toolbars, and templates to generate common expressions in standard ways, using content management systems to reuse common equations that have been thoroughly checked, employing filters to ensure that very similar-looking characters are not inadvertently being substituted, and so on. One conversion vendor told us that they actually have each equation marked up twice by independent workers. They then compare the two to automatically flag potential problems. For any of these strategies to be successfully implemented typically requires a staff person with deep MathML knowledge, who is capable of setting policies for the organization as to what "good" MathML is.

4. Production challenges

In production, perhaps the chief challenge is coming up with a strategy for managing style and fine-grained layout issues, while preserving good separation of style and content. What is the right balance between enforcing a house style, and hand tweaking problem equations? How can you minimize situations where an equation looks perfect on screen, but lousy on paper? What should you do about Web output? What are the options for making mathematics accessible on the web for sight-impaired individuals?

4.1. MathML support in composition engines

The production phase of XML+MathML workflows exhibits the widest variation. Mostly this is because requirements vary widely, and there is a wide variety of tools and approaches available to serve widely varying needs. One common situation arises when there are complex document-level formatting requirements. Typically, these workflows will use a dedicated composition engine, such as Epic E3, XPP, or 3B2. Most composition systems in this category have some level of XML support, and a few have some level of native MathML support. The systems that currently have native support however, currently only have partial MathML implementations, so accessing the full layout power of the composition engine typically requires mixing some sort of system-specific markup in with the MathML. For example, many XPP typesetting directives are only accessible via processing instructions inserted into the MathML. Since such constructs can cause problems with some MathML tools, it is important to make sure the MathML copy editing tools and the composition engine work together.

Native MathML support in composition engines is still the exception, though. Thus, another common workflow pattern is to separately process MathML into images (vector-based when quality is paramount, raster otherwise) and then re-integrate the images into the final output. This has the appeal that almost all composition engines can support math in this way, but the downside is that it is complex to set up, and it can be very tricky to get high-quality output, since

generally images do not integrate perfectly inline with text. Baseline alignment, fonts, image resolution, side bearings and so on can all present quality problems.

4.2. Updating publications on a regular schedule

Another common workflow pattern, particularly in enterprise settings, arises when output formatting is fairly regular and not too complex (e.g., regular updates to technical manuals). In these situations, the emphasis is usually on coming up with a robust, high-volume, hands-off composition process. Such workflows are more likely to rely on techniques such as using XSL to produce HTML, or XSL-FO engines. While this leads to rather different choices for document-level composition engines, the choices for handling the math are much the same – go with an integrated solution such as Antenna House's recent XSL-FO formatter with native MathML support[8], or preprocess the MathML into images which are then reintegrated into the document output.

4.3. Maintaining consistent style—tools and techniques

Whether an organization opts for an integrated composition engine, or preprocessing and reintegration of math as images, all workflows must come up with a strategy for maintaining a consistent typesetting style within and across documents, as well as a strategy for dealing with differences in equation formatting between WYSIWYG editing tools and final output. In workflows aimed primarily at a single output medium, the simplest solution is to use the same math rendering engine for both copy editing and final composition. However, this may not be possible, depending on the choice of document editor and composition engine. Moreover, it runs the risk of encouraging people to rely on pure visual editing, ignoring the underlying XML structure. While this may be the path of least resistance in the short term, it obviously leads to data that is less reusable and that capitalizes less on the efficiencies of uniform processing that one expects from XML.

In situations with multiple output media, it is less likely that the same math renderer will be appropriate in all cases, or at least not with the same settings. At the most superficial level, rendering must take place at much higher resolution for print formats, which may involve slight differences in layout algorithms. Similarly, workflows that use a dedicated composition engine such as XPP for print typically need to add renderer-specific processing instructions for print composition, but these instructions typically interfere with rendering in HTML. Consequently, a common strategy is to use a transformation technology such as XSLT to preprocess XML+MathML for a specific output medium and renderer. Thus, one might have an XSL stylesheet for HTML output and another for print output.

The use of XSL or similar technologies to preprocess for a specific renderer has the added side-benefit that the same preprocessing pass can add all kinds of formatting that would not be appropriate for the archival source, such as choices of specific fonts, etc. This is a very powerful technique, and provides organizations with a level of centralized control over style. However, it typically requires particularly "clean" source files and access to XSL expertise to create and maintain stylesheets. Clean source files, in turn, require well-trained editorial staff that can deal with an edit-compile-preview work process.

5. XML+MathML in practice

In the real world, requirements vary widely and no two workflows are the same. By identifying and understanding the problem areas others have confronted and overcome, you can greatly increase your chances for success.

5.1. American Institute of Physics

AIP is a publisher of scientific journals for a number of professional societies. They outsource conversion of manuscripts to XML. They use Arbortext Epic with the MathFlow Editor for copy editing. For print composition, they use XyEnterprise's XPP. XPP processing instructions are added by a custom in-house transformation process, and are hand

[8]Antenna House is the publisher of XSL Formatter V3. http://www.antennahouse.com [http://www.antennahouse.com/product/axfo30/axfo3top.htm].

tweaked by specialists. When Web output is required, they use a custom in-house process involving a Perl transformation and images for the mathematics.

5.2. Airbus S.A.S.

Airbus publishes a large volume of technical manuals. Authoring is distributed throughout the enterprise and content is accumulated in a content management system (Documentum). Final assembly and editing is performed with Epic and MathFlow. Composition is performed by using the WebEQ Equation Server to preprocess MathML into images.

6. Conclusion

The lessons learned from those who have gone before are usually quite valuable. Although no two workflows are alike, it's easy to see similarities and benefit from the success stories. All of the examples provided above are ones we have gathered from our clients and partners, who have gone unnamed for the most part. The common thread is that XML+MathML has truly come of age, and is proving itself every day in workflows around the world.

Bibliography

[MathML] *Mathematical Markup Language (MathML) Version 2.0 (Second Edition) [http://www.w3.org/TR/2003/REC-MathML2-20031021/]*, David Carlisle et al.,Worldwide Web Consortium,2003.

Biography

Bob **Mathews**
Director of Training
Design Science, Inc. [http://www.dessci.com]
Long Beach
California
United States of America
<bobm@dessci.com> [mailto:bobm@dessci.com]

Bob Mathews is the Director of Training for software developer Design Science, Inc. A former career military officer, he has served as an Instructor Pilot in T-38 and KC-135 aircraft, as a Flight Commander, and as Deputy Chief of the Strategic Air Command Instrument Flight Course. Transitioning from the stratosphere to the lower atmosphere, Bob spent the next several years teaching high school mathematics and serving as department chair.

With an undergraduate degree in Electrical Engineering and a graduate degree in Management & Human Relations, Bob enjoys teaching and has taught in such diverse environments as the cockpit of a supersonic jet, a high school mathematics classroom, a corporate boardroom, and a Sunday School class. He has led seminars and presented papers at conferences on three continents, and is often an invited speaker at math teacher conferences and software user group meetings. His current interests include using a standards-based approach to technical publishing and designing interactive math applications for web pages and distance learning environments.

Robert **Miner**
Director of New Product Development
Design Science, Inc. [http://www.dessci.com]
Long Beach
California
United States of America
robertm@dessci.com [mailto:robertm@dessci.com]

Robert Miner received his college education and subsequent graduate and post-graduate training at the University of Maryland, Oxford, and Universitat Bern. He focused on mathematics, receiving a Ph.D. in 1991. In 1995, after teaching for four years at the University of Oklahoma, Dr. Miner decided to move to the Geometry Center at the University of Minnesota where he became involved in the World Wide Web consortium initiative to standardize an XML markup language for mathematics. He eventually co-chaired the technical working group that developed MathML. Dr. Miner and two other researchers spun-off a company to commercialize the software and web content developed at the Center, which was acquired by Design Science in 2000. Since then, Dr. Miner has worked to develop the MathPlayer and MathFlow products, written and spoken extensively on the impact of MathML on technical publishing, and initiated a research program on adding value to electronic math content, including an NSF research grant awarded in 2003 to develop math-aware searching.