

Casting a critical eye on the Next Big Thing in technical publishing.

XML, Growing Up Fast

BY SARAH S. O'KEEFE, Associate Fellow

This year marks the ten-year anniversary of XML. In 1998, technical content development was generally based on a desktop publishing paradigm. The best practice for efficiency in publishing was to single-source content—perhaps by developing in FrameMaker and outputting HTML content through WebWorks Publisher. A few very large companies had expensive systems based on SGML, or Standard Generalized Markup Language (the precursor to XML).

Today, the state of the art is an XML-based publishing environment, with push-button delivery of various output formats, often in multiple languages. XML has found its way into organizations of all sizes. A lot has changed in ten years.

Nonetheless, I believe that the current impact of XML is small compared to what's coming in the next decade. The evolution of content creation from format-driven publishing to structured authoring is a paradigm-shifting (and cliché-ridden) transition.

While wading through a swamp, it's important to focus on each step to avoid getting muddy. But as a

result, we forget to see the bigger picture—the wading birds, the snake sunning itself, and the marsh flowers. So in this column, I want to set aside the pressing current issues—how to convert badly formatted legacy content to XML, whether DITA is the right choice or not—and instead look forward five to ten years.

Note: STC provides a forum (stcforum.org/viewforum.php?id=51) where you can discuss this and other *Intercom* articles. If (when) you disagree with what I write here, you can start a discussion there. You can also contact me directly using the information at the end of this column.

Solving the Printing Problem

Producing attractive printed output from XML is still unbelievably difficult. In the original version of my “XML and Structured Authoring” white paper, published in 2002, I blithely assumed that the challenges of creating print layouts through the Extensible Stylesheet Language Formatting Objects (XSL-FO) would be solved “within five years.” It is now six years later, and the XSL-FO challenges remain.



One problem is that beautiful print layouts almost always require formatting exceptions. When fitting copy to a page, a production editor might tighten line spacing imperceptibly to fit an extra line or two on a page, which then causes the remaining pages to fall into place nicely. The automated tools cannot match this level of sophistication, so automatically generated layouts have a less polished look than a layout that was fine-tuned by a production editor.

Many organizations no longer print documentation for their customers. Instead, they provide PDF files, which customers can print themselves. Thus, I believe that the print problem will be solved in two ways:

- Lower standards. Organizations decide that the printed output from an automated XML-based workflow, while imperfect, is good enough, especially if Web-based help is the primary output.
- XML invades the page description business. Adobe is working on a project, code-named Mars, that will create a version of PDF written in XML. This will eliminate the need for processing XML through XSL-FO, InDesign, FrameMaker, or other XML-aware publishing tools. Instead, we could use XSL transformation to produce “PDF-ML” from XML source content.

It's also possible that we will finally see some progress in support for XSL-FO creation tools. However, the software vendors do not appear to be focusing in this area. The vendors that make development tools are more focused on XML and XSL, which are used much more widely. Another possibility would be that a publishing tools company, such as Adobe, Quark, or PTC, would produce a graphical XSL-FO editor. However, these companies appear to perceive XSL-FO as a threat to their existing print solutions—why produce something that competes directly with InDesign/FrameMaker, QuarkXPress, or the E3 publishing engine?

Convergence Continues

As documentation moves online and becomes interactive, training is moving online and becoming less interactive. The distinction between e-learning and interactive documentation is blurring.

Automation in this area is still limited because graphic, sound, and video formats tend to be binary. They require proprietary authoring tools, and the file formats are difficult (or impossible) to manipulate programmatically. The Scalable Vector Graphics (SVG) format, which describes vector images in XML, can provide a foundation for vector graphics in XML. But photographs and screen

Over time, XML will become as ubiquitous as HTML.

captures do not yet have an equivalent format.

Flash authoring is moving (slightly) in this direction. According to Colin Moock (a Flash consultant and author), the next version of Flash (CS4) will use a new format, XFL, to write out Flash files. Currently, it uses Flash (.fla) format, which can only be read by Flash authoring tools. XFL will be XML-based—at least for text within the file and for the description of the assets (sound and video) that are included in the file. I assume that this means that the sound and video files will still be in a binary format.

The XFL format will allow us to use text stored in XML as the foundation for content delivered in Flash as well as Web and print deliverables. This opens up Flash authoring to single-sourcing techniques for the first time.

Sound and video use numerous standards, and streaming technology adds another layer of complications.

Imagine a multimedia development environment where all content is encoded in XML. We would be able to apply the same automation we're developing for text today to graphics, sound, and video.

I can already hear the protests about how XML is a terrible, inefficient format for encoding anything other than text. Remember that the same argument was made that XML was horrible for anything *including* text. Yes, it's inefficient, but it's not going to matter because the hardware will be able

to handle it. When SGML was developed in 1979, it provided "tag minimization" options because this made the text files smaller. At the time, bringing file size down a few kilobytes made an appreciable difference in processing time. By 1998, tag minimization was omitted from XML because the file size was no longer a problem for the computer hardware. XML files are more verbose than SGML files, but easier to parse. Over time, textual and graphical content has evolved from binary to compacted text formats to verbose text formats. I expect that audio and video will follow the same path.

In addition to the technical aspect of convergence, where information is migrating into XML, we also have an interesting convergence of "professional" and user-generated content. I plan to address that in a future column.

The Dream of Better Authoring Tools

Some of you may remember the early days of HTML, when we wrote content in Notepad. The first generation of HTML tools wasn't terribly good, either. But today, we have Dreamweaver, which is the de facto standard for Web authoring and design. Why is it so successful? The software reflects a deep understanding of what Web content creators do, and it provides features to accomplish those tasks efficiently and elegantly.

Unfortunately, we have nothing comparable for XML content creation. In one segment, we have developer tools started up to support XML-based content in addition to XML data. Other tools use a desktop publishing foundation with structured authoring support. A third group evolved from SGML authoring to support XML authoring. But the purpose-built XML

authoring tools are not mature yet—nobody offers the elegance and clarity of purpose for XML that Dreamweaver does for HTML.

So while I'm making predictions that may or may not come true, I'd like to outline the list of software requirements for a great authoring tool. It should include the following:

- Features should address not just authoring tasks but also the publishing and production side. This might require integration with content management systems, or the ability to send a completed document onward in your workflow.
- There should be several ways of viewing and editing content, including plain text (code view), tree view, word processor view, and output view for every format that you produce. Content should be fully editable in all of these views.

Figure 1. When Flash understands XML, we can link XML-based text into Flash files.

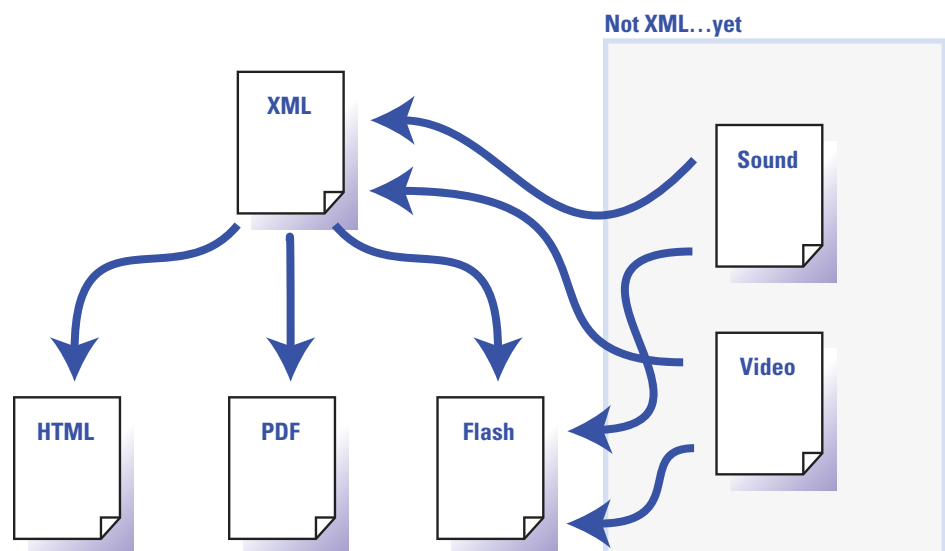
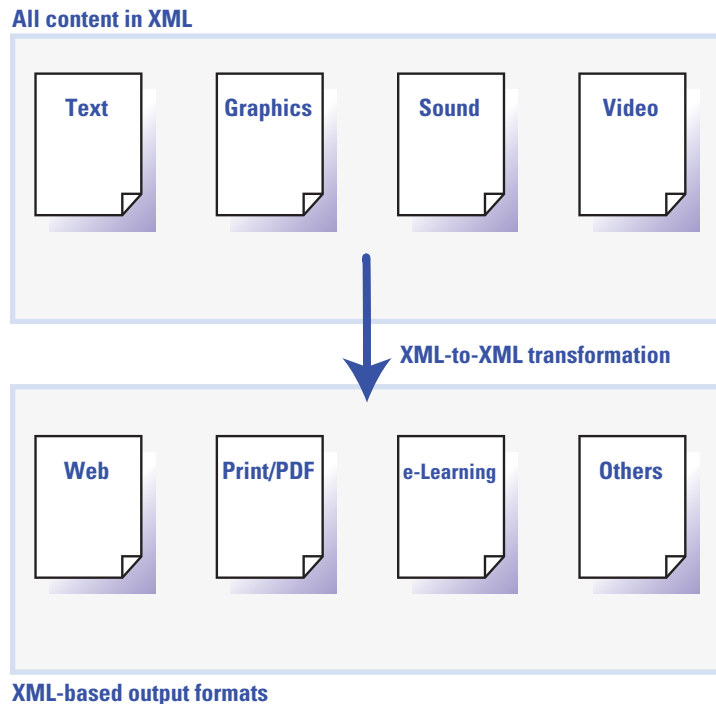


Figure 2. One format to rule them all...



- All of the editable views should support conditional content display, so that you can author in a preview mode for a specific output or with all content showing.
- All of the editing views should be able to display all supported document components, including text, graphics, audio, and video. For example, a hypothetical iPhone content mode should (currently) not render any Flash video content. PDF mode should show all multimedia content. Print preview mode should replace live video with static graphic images and disable hyperlinks.
- Assigning metadata to a particular content unit should be fast and easy.
- As you author content,

the editor should offer intelligent suggestions on where reuse might be appropriate. (Interestingly, AuthorIT and Flare are both moving in this direction. Sadly, their authoring tools are not XML authoring tools.)

As I write my list of demands, it occurs to me that I should defend the software vendors. The reason that nobody has built a tool like this yet is because it's hard. The technical documentation market is not known for spending freely on software, even when the productivity gains are compelling.

Living Up to Hype


Many of you are probably tired of hearing about the glories of XML. Perhaps you like your current feature-

rich, elegant, non-XML authoring environment and wonder why moving to XML requires a huge step backward in usability for authors.

It's my opinion that over time, XML will become as ubiquitous as HTML. The structured authoring paradigm, which allows us to enforce required document structures, is compelling for technical documents. Software will eventually address the authoring and publishing challenges, and more content types will migrate to XML formats. The implementation cost will decline and the value of an XML-based environment will increase, resulting in more organizations migrating to XML.

If you are considering XML, develop your own

requirements list. If highly designed print layouts are mandatory for your content, XML may not be ready for you. On the other hand, if the convergence of documentation and training is of interest to you, you may want to consider XML sooner rather than later. The structured authoring paradigm is challenging for many authors, and if your authors have had difficulties mastering the easier unstructured tools, the state of XML authoring tools may present a significant obstacle.

We live in interesting times. 

SUGGESTED READINGS

ChemicalML discussion, cml.sourceforge.net/historical/position.html

Colin Moock's blog post about XFL, www.moock.org/blog/archives/000269.html

MathML, www.w3.org/Math/

Project Mars, labs.adobe.com/wiki/index.php/Mars

SVG, www.w3.org/TR/SVG/

Sarah O'Keefe (xmlstrategist@scriptorium.com) is founder and president of Scriptorium Publishing Services, Inc. (www.scriptorium.com), based in Research Triangle Park, North Carolina. She is a senior member of the Carolina Chapter STC and of the Consulting and Independent Contracting and Management communities.