

XML & Lone Writers:

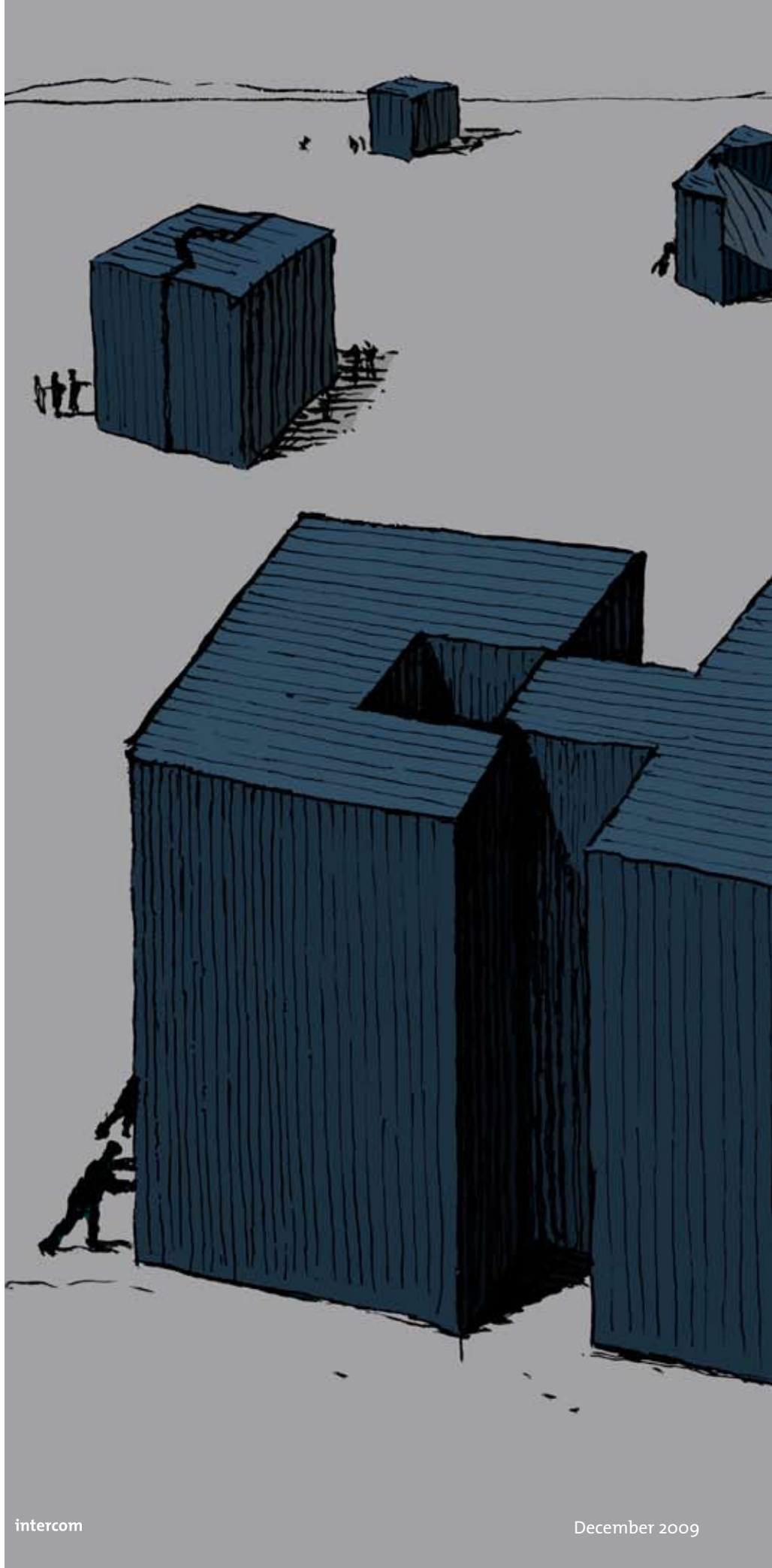
BY SARAH O'KEEFE, *Associate Fellow*

A lone writer is someone who works as the sole technical writer in an organization. With approximately 1,000 members, the STC Lone Writer Special Interest Group (www.stc-lone-writer.org) includes nearly 10 percent of the overall STC membership. As a soloist, a lone writer often has great control over writing tools, technologies, and process—provided that the chosen approach fits within the generally nonexistent documentation budget.

According to our industry research, lone writers are much less likely to be working in XML than writers in larger authoring groups. In a 2009 survey that Scriptorium conducted (see Resources at the end of this article), approximately 30 percent of respondents overall were using structured authoring, compared to less than 20 percent of lone writer respondents.

In September 2009, I consulted members of the Lone Writer SIG mailing list and asked about their interest in XML. LaKisha McKenney of Meridian Knowledge Solutions summed up the issue nicely:

I think I'm on the fence about whether XML/structured authoring is useful/valuable for lone writers, mainly because of the time it would take to implement it, and whether that would outweigh the benefits I (or others at our company) would get from it.





Can They Go Together?

The Value of XML for a Small Group

The advantages of an XML-based workflow for a larger group of writers—whether 50 or 15—are clear. But what about a lone writer? Many of the justifications for implementing XML still apply when you *are* the writing department:

- **Consistency.** It's easier for one person to organize information consistently than it is for a group, but it's still nice to have the automatic verification that XML provides. The enforcement of required structure can take the place of the production editor that you wish you had.
- **Efficiency.** Instead of formatting each document as you go, you set up the formatting once and then generate your output. Formatting is fast, reliable, and automated. This can be very helpful when you face a deadline with no one to assist you.
- **Reuse.** Assuming that you need to reuse content, the XML framework can help you do so efficiently. You can go beyond what's possible with conditional text or build tags and automate much of the effort.
- **Content exchange.** You can use XML to reduce the amount of friction involved in transporting content from one system to another.

But as McKenney explained, the implementation effort is a huge obstacle. Many larger authoring groups assign a team to evaluate, implement, and maintain a structured authoring environment. A structured authoring environment for a lone writer must meet the following criteria:

- **Low cost.** Budgets for lone writers are generally small to nonexistent.
- **Low-impact implementation.** A lone writer must continue to deliver while building an XML workflow. There is no “implementation team” option.
- **Low maintenance.** Along with the nonexistent implementation team, lone writers need to maintain their environment without access to a maintenance team. Minimal maintenance is critical for small groups.

Using XML to Import Information Into Your Current Workflow

XML provides an efficient way to exchange information among otherwise incompatible tools. For example, Fei Min Lorente, a lone writer at the Medical Division of ON Semiconductor, uses XML for code comments. She explained, “A few hundred pages [of documentation] are generated straight from the comments in the code, and the code samples (which can be compiled and verified) form the code examples in these reference chapters.” She uses scripts that create XML from the comments in the source code, and then opens the XML in structured FrameMaker where it's “instantly formatted.”

If you have source information outside your current authoring environment, consider whether you could use XML to bring information into your documents automatically. An automated approach is more efficient than manual updates and is also more accurate (no more copy-and-paste errors). In addition to code comments, this approach can work for product specifications

(often stored in a database), parts catalogs, software error messages, and other reference information.

Because many common authoring tools support XML import, you may be able to add XML-derived content to your current environment without actually moving to XML-based structured authoring. (Lorente's approach uses a structured authoring tool, but you could do something similar with AuthorIT, MadCap Flare, and even Microsoft Word.) If you are currently bringing in content manually or spending lots of time on manual reformatting, an XML-based import process could save you a significant amount of time.

Quick Start

- Assess the XML import capabilities of your authoring tool.
- Assess the XML export possibilities for the content source (a database or source code).
- Generate XML from the content source that conforms to what your authoring tool can accept (if necessary, use XML-to-XML transformation as an intermediate step).
- Configure the authoring tool to import XML content and apply standard formatting to it automatically.

A Lone Writer XML Roadmap

XML is not right for every writing environment, but I am going to assume that you have looked at your situation and decided that you could benefit from XML for increased consistency, reuse, and efficiency.

1 Research and Learn

The first thing you need to do is to get comfortable with the XML ecosystem. Read up on XML and structured authoring, get to know the DITA (Darwin Information Typing Architecture) standard, and so on. Download an XML authoring tool, such as Serna Free XML Editor (www.syntext.com/products/serna-free/), and learn to use it. Understand terms such as "validation," "DTD," and "content model." Learn basic XSLT programming.

Once you move to XML, you will not have a safety net, so spend some time in

Step 1. You need to be your own XML expert.

2 Analyze Your Current Workflow

Once you have a reasonable understanding of the advantages and disadvantages of XML, take a hard look at your current content creation process. Can XML help you fix problems in your workflow? For example, if you spend lots of time correcting step numbering and fixing formatting, automated formatting could help. But XML is not going to help you fix a poor working relationship with the product development team. Collaboration requirements are also a potential roadblock. Janet Swisher, while employed as a lone writer at Enthought, Inc., decided against XML in that environment because she needed input from others. "As a lone writer I depended more than many writers on collaboration with non-writer engineers, and therefore needed to use least-common denominator tools that were accessible to them," she said. "They were not interested in learning another markup language."

Are you spending a lot of time copying, pasting, and reformatting print content for online delivery? Perhaps you could be more efficient with automated reuse and formatting.

If you can identify a few major problems that an XML-based workflow could solve, it's time to move on to Step 3.

3 Build Your Business Case

As a lone writer, it's unlikely that you'll be presenting a formal business case to management to get your project funded. (After all, you probably won't be getting any funding!) But you need to be able to justify the time and effort you are going to put into additional XML work. So, your business case can be a very simple set of bullets. For example:

- Amount of time spent on inefficient, old process (copying and pasting, reformatting): x hours
- Amount of time spent on efficient, new process (automatic reuse, automated formatting): y hours
- Amount of time saved: z hours = x hours - y hours

If z is a negative number, stop. You do not have a business case. If z is a positive number, you still need to factor in the amount of time required to build your XML-based environment, but you do have a chance.

At this point, you probably need to build at least a high-level estimate for implementing XML so that you can figure out whether the effort is worthwhile. This is, of course, quite difficult to do without knowledge of your environment (which you have) and prior experience with XML implementation (which you probably don't have).

Consider reverse engineering the XML implementation number. Hypothetically, let's say that you've calculated that XML can save you 50 hours of effort per deliverable and you have four deliverables each year. That means you could potentially save 200 hours per year. If you can implement XML in 200–400 hours, you have a reasonable business case for moving to XML, as you will start to see savings after two years. Armed with that knowledge, you can figure out how much effort to put in the remaining steps in this roadmap.

4 Build a Small Prototype

Prototyping lets you try out the workflow you plan to use. A successful prototype should be as small as possible, but big enough so that you can get a realistic sense of whether the approach would work in the real world. Here's what you need to do.

Choose an XML architecture. You're probably going to want to use DITA. It's well supported, it works for topic-oriented information, and it's easier than building your own structure. If you need a custom structure, proceed with extreme caution.

Choose an authoring tool. Your ideal authoring tool should be something you are comfortable using, produce valid XML content, and be within your software budget. Definitely consider Serna Free and oXygen XML Editor (www.oxygenxml.com) before moving up to more expensive tools.

Start creating content. You can migrate content in several ways, but try to match how you will create

information in the real world. For example, if your organization is constantly creating brand-new products, start writing from scratch. If you generally use version 1 of a manual to create version 2, match this approach in your prototype. Explore some options for migrating content efficiently, but for now, get just enough information into your new system as quickly as possible. Try out the features you intend to use in a production environment, such as various techniques for reusing information or for labeling information as conditional.

Customize the output to meet your specifications. If you are working in DITA, you will likely need to learn the DITA Open Toolkit. Start by customizing the HTML output. At a minimum, you'll need to modify the CSS file that controls formatting and insert your corporate logo and/or copyright information. Keep things as simple as possible. If you are going to need cross-browser, cross-platform help, look into publishing your XML content through a help authoring tool. Once you have the HTML working, look into PDF. Customizing PDF through the Open Toolkit is extremely challenging. If you can avoid it (for example, by publishing through FrameMaker), consider that approach.

At this point, you should have some content in XML and the ability to create basic output. You can now evaluate your prototype system. What refinements are still needed? Does the system solve (or hold the promise of solving) the problems you identified in Step 2?

After getting this far, you may decide that the difficulty involved in moving to XML exceeds the benefits. If you are going to kill the project, now is the time. Do not rush this decision.

5 Full Implementation

If you decide that you're on the right track with the prototype, you can move forward to full implementation. You'll want to fix any issues that you deferred in the prototyping process, make a decision about authoring and publishing tools, and generally refine your environment.

You're also going to have to make some potentially difficult decisions

about migrating unstructured content to XML. You have three basic options:

- **Migrate everything.** Convert everything to XML so that you can use it in the new environment.
- **Migrate nothing.** Write everything from scratch to avoid having to re-write content.
- **Migrate content selectively.** Convert content you believe you can use in your new documents. Or, consider maintaining some projects in the unstructured environment and migrate only those for which the XML approach provides the maximum benefit.

During implementation, you'll want to get all of your tools working. In your spare time, make sure you provide at least some basic documentation on how the system works. As John Sgammato, a lone writer at Imprivata, Inc., pointed out, "Lone writers work without a net. If I get hit by a bus, my employer will have a hard time finding someone who can figure out this Rube Goldberg system of mine." It would be a good idea to address the "hit by a bus" scenario with some system documentation. I haven't been hit by a bus (yet), but I have often referred to my own documentation because, a year later, I couldn't remember why I chose a particular solution. Documentation is also useful for your own sanity.


You Are on Your Own

A couple of lone writer respondents asked me to discuss how to justify a consultant to help with XML implementation. Although this could be highly beneficial to them *and* me, I have to point out that it is unlikely that an organization that employs a lone writer will cough up funding for extensive consulting. Any request for outside support, then, should be highly focused. For instance, consider asking for consulting assistance only to review an implementation plan that you have already written. Or, look at bringing in a consultant to address highly technical tasks (such as DITA-to-PDF conversion) for which the learning curve is steep.

If you feel that implementation is impossible without a consultant at your

side, go back to your business case. What is the business justification for implementing XML? How much would your organization be willing to pay to get the shiny new features that XML offers? If you are looking at significant return on investment or cost avoidance, you may be able to get a consultant in the door. (But I seriously doubt it.)

Conclusion

The relatively low percentage of lone writers who have implemented XML is a logical result of the typical lone writer working environment. Although it *is* possible for lone writers to implement XML, a very cautious evaluation of the idea is definitely in order. Given the current status of the authoring and publishing tools, any lone writer who implements XML will need to master fairly demanding tools and technologies. If you are the type of person who avoided HTML until nice, friendly visual editors were available and you are lost when looking at HTML tags, XML is not for you. Wait a few years for improvements in the available tools. If, however, XML can provide compelling advantages to your workflow, don't let your status as a lone writer stand in your way. 

RESOURCES

O'Keefe, Sarah. "When is XML the Wrong Answer?" *Intercom*, November 2007.

———. "The Hidden Cost of DITA." *Intercom*, April 2008.

O'Keefe, Sarah, and Alan Pringle. "The State of Structured Authoring in Technical Communication." 2009. Available for purchase from www.scriptorium.com/books/the-state-of-structured-authoring-in-technical-communication.

Sarah O'Keefe (xmlstrategist@scriptorium.com) is founder and president of Scriptorium Publishing Services Inc. (www.scriptorium.com). Sarah is an STC Associate Fellow and member of STC's Carolina, India, TransAlpine, and UK Chapters and of the Consulting and Independent Contracting, Europe, and Management SIGs.