

Using XML

**A look at XML as a document
authoring and management tool**

Scott Nesbitt

Copyright (C) 2000 - 2004 by DMN Communications

The Problem

Contrary to what some people believe, writing documentation isn't simply a matter of waving a magic wand and having manuals appear out of thin air. And documentation is about more than sitting at a keyboard and typing. What many fail to realize is that technical writing involves research, interviewing developers, writing, editing, and formatting documents.

Technical writers are also concerned with:

- Efficiently maintaining the documentation they produce
- Tracking and managing revisions
- Taking elements from a single document source and using them to build printed manuals, online Help, and electronic documentation for posting on the Web or for publication on a CD-ROM

The information technical writers need comes in many forms, including overview text, procedures, images, code samples, and more. Traditional methods of creating multiple output from one document, like the venerable "cut and paste" technique, are too cumbersome and labour intensive. Using a conditional text feature, as found in *FrameMaker*, is a better solution. However, it takes time to set up and use. And not every authoring tool has this kind of function.

Writers also need a way to quickly find information within various documents. The information may be buried anywhere, and it isn't useful unless writers can get at it when they need it. Most technical writers have been in a situation in which they know the information exists, but don't know exactly where it is. It's annoying, frustrating, and time consuming to track these bits and pieces down. Time is something technical writers on a deadline rarely have.

Enter XML

XML stands for Extensible Markup Language, which is used to create documents that contain *structured* information. This structured information is made up of both:

- Content, which is the text, images, etc. that make up a document.
- Information on the structure of a document. This information delineates the role of each piece of content – for example, the difference between a paragraph and a numbered list. It also rigidly defines the order in which content can appear – for example, a chapter title cannot appear within a paragraph.

If you have experience with markup languages like HTML or SGML, then XML will seem familiar. XML uses HTML-like tags to structure a document. However, unlike HTML, XML has no real tag set. As its name states, XML is extensible – you can define your own markup and create customized tags to provide functionality lacking in HTML. If you want to specify a first-level heading, you can use the HTML tag `<h1>` or `<heading1>`. A full discussion of XML is beyond the scope of this report. For good introductions to the language, visit XML.com or XML101.com.

XML is concerned with content and the structure of content, rather than formatting details like typefaces, indentation, etc. And it is this focus on structure that makes XML perfect for use in documentation, especially in authoring and managing the component pieces of information that make up those documents. More importantly, XML is well-suited for *single sourcing* documents. Single sourcing “involves identifying all information requirements up front, then developing them from a single source.”¹ The single sourcing method developed by the Rockley Group (a Toronto, Canada-based documentation firm) involves lumping all information into one file, then breaking the information out into various formats as needed. So, what goes into a user manual will differ from the content of a product overview or Help system.

You can think of an XML document as being a tree, with delineated branches and offshoots. For example, the body of the document (“the trunk”) can contain clearly-defined sections (“branches”). Each section can contain subsections and passages marked up for specific purposes (for example, sample code). While the branches and offshoots make up the document as a whole, you can also access and edit them independent of the entire document.

The Software

The first step in using XML is to learn about the language. There are a number of resources available in print and online. See Appendix 2 for details. The next, equally-important step is to choose the right software. There are a number of XML-based document creation and management systems on the market, and there are more are on the way.

Authoring Software

You can author in XML using a number of applications, including:

- FrameMaker version 7.0
- ADEPT*Editor
- XMetaL

¹“Single Sourcing Whitepaper”, Hackos, JoAnn and Anne Rockley (July 14, 1999), SingleSourcing.com

-
- XMLmind XXE
 - oXygen
 - EpcEdit
 - XEmacs, with the psgml or psgmlx packages

Many of these applications enable you to create and edit documents in familiar word processor-like environment. Others are text-based applications with specific XML authoring features. Both types of applications have their strengths and weaknesses. The choice of tool you make will depend on how comfortable and efficient you and the members of your team are with using WYSIWYG or plain text tools.

Document Management Software

Some of the better XML-capable document management applications include:

- ArborText *Epic*
- XyEnterprise *Content@XML*
- Documentum *4i*
- Lightspeed *Astoria*
- Author-IT
- Live Linx
- Borges
- SiberSafe

Not surprisingly, these applications have similar core functions – collaboration, re-use of information, and version control. Others have specialized features like the ability to map out and track workflows, as well as electronic review.

Most of these programs also integrate with popular authoring tools like *FrameMaker* and *ADEPT*Editor* (some handle Microsoft *Word* files as well), and can generate output in several formats. Most are also fairly expensive, but you get what you pay for. In most cases, you are getting quite a lot.

Of course, you're not limited to expensive commercial offerings. Some shops use change tracking and version control tools like CVS and Perforce. If you are interested in lower-cost alternatives like these, visit SourceForge and do a search for content management.

Choosing a tool isn't as cut and dry as it may seem. You have to decide what your needs are, then investigate as many applications as you can. You also have to take into account the authoring tool you are currently using. If it is not supported by the XML system you have chosen, there will be an additional expense in purchasing and learning a supported application.

Chances are, you won't be creating documentation from scratch; you will have an existing documentation set. If you are in the position of starting from scratch, you can start authoring in XML. For existing documentation, *FrameMaker* is the most widely-supported application, although certain tools only integrate with *Word* for Windows.

That's Nice, But ...

... what does all of this mean to you? Simply put, the tree-like structure of an XML document allows you to easily manage your information. And you can chunk the information to quickly extract what need when you need it. Not only that, you can take the information you need from a document and use it elsewhere. For example, if you are building an online Help system, you can extract all the relevant procedures but leave out any superfluous overview information. You can do this using, say, the conditional text feature in *FrameMaker* but doing so can be awkward and time consuming. On top of that, using XML enables you to re-use standard chapters – introductions, glossaries, “how to use this program”-type sections, etc. – quickly and easily.

What Now?

You have the content. You have the tools. Now what do you do? One of the first things is to create a functional specification². The functional specification will literally maps out what your manuals will contain and how you intend to structure the information in the manuals. And that leads to the next step ...

... analyzing your documents

Your analysis will help you:

1. Decide on a sufficient level of granularity in the documents. You must decide how much granularity is enough, based on your present and future needs
2. Determine how to order the elements in the documentation. You must consider *every* way in which elements nest

²See chapter 7 of *Mastering XML* by Linda Burman, Chuck White, and Anne Navarro for more information.

-
3. Determine which document elements you want to keep, and which you want to discard
 4. Decide how to tag these elements

Putting It All Together

Once you have decided how you want to structure your content, you must now get a DTD. DTD stands for *Document Type Definition*, which is a file associated with an XML document that specifies how to present the elements that make up the document. You can:

1. **Create your own DTD**, basing it on the styles used in your standard template.
2. **Use an existing DTD**, like DocBook.

If you are feeling ambitious, then by all means create your own DTD. As you will find out, creating a DTD isn't a trivial task. On top of developing a DTD, you must also select an approach to take to designing the DTD. These approaches include:

1. **Chief Engineer approach** – having a single XML expert create the DTD
2. **Facilitated Team approach** – having a project team create the DTD with the help of an XML expert
3. **Informed Partnership approach** – the XML expert creates the DTD with guidance from the project team³

The main advantage to creating your own DTD is that you can make the elements in your DTD mirror the tags in your current template. The hope is that you won't have to do any tweaking to get things to work properly.

The advantage of using an existing DTD is the ability to “take advantage of design, development, testing, and documentation done by others”⁴. Using an existing DTD saves a lot of effort and resources, but it may require some work to adapt your document template to the DTD. Using an existing DTD can eliminate a lot of the time it takes to develop and test a DTD.

The easiest and most logical choice for an existing DTD is DocBook. DocBook is a very complete DTD intended for software documentation. It was originally created for SGML, but also has an XML version. However, according to *DocBook: The Definitive Guide* the XML

³See chapter 7 of *Mastering XML* for a detailed discussion.

⁴*Mastering XML*, pp. 154.

version “hasn’t been officially adopted by the OASIS DocBook Technical Committee yet.”⁵ On top of that, you will probably have to modify your template to match the tagging specified in DocBook. For example, in an API reference created in *FrameMaker*, you might have a style called `code` or `codesample`. In an XML-authored document, you would use the DocBook element `programlisting` instead of your *FrameMaker* style.

Because XML is more concerned with content than style, you will need a way to apply formatting to your documents. That is where XSL (Extensible Stylesheet Language) comes in. XSL is a method of separating style from content in XML documents. In short, XSL acts like the templates you use in word processing or desktop publishing software. To use XSL, you must create an external file linked to your documents. The file contains the XSL information that controls the appearance of each element. The external file means you can use the same styles for multiple documents without duplicating your work. For more information on XSL, visit the World Wide Web Consortium XSL site.

XSL is a very intricate language. Some have compared it to a low-level formatting language. As it stands, XSL “is a bit of a moving target right now ... the XSL specification is moving in a completely new direction”.⁶ If you’re familiar with HTML, you can use Cascading Style Sheets (CSS) to format your XML documents.

The CSS you use with an XML document differs slightly from what you would use with an HTML file. Take the following fragment of a style sheet:

```
para {display: block;
      font-family: sans-serif;
      font-size: 11pt;
      text-align: left;
      margin-bottom: 13pt}

subhead {display: block;
         font-family: serif;
         font-size: large;
         font-weight: bold;
         margin-bottom: 10pt;
         text-align: left}
```

Notice the `display:block` and `margin-bottom` attributes? You wouldn’t normally use them with a style sheet for an HTML document. However, they are essential with an XML file – if you don’t include these attributes, your document will appear as one large block of text.

You should treat CSS as a stop gap solution to formatting your XML documents. It lacks the flexibility and power of XSL. But until XSL comes into its own, CSS is good enough.

⁵Walsh, Norman and Leonard Mullener, *DocBook: The Definitive Guide* pp. 575.

⁶Eckstein, Robert, *XML Pocket Reference* pp. 575.

Note: If you need more information on CSS, you can find it a numerous Web sites including Web Developer's Virtual Library and WebReference.

Conclusion

XML can change the way in which you work with documentation. And not only in the way the documentation is created and managed. By moving to XML, there will be a fundamental shift in the way you view documentation. You will move from seeing manuals and Help files as documents with a beginning, a middle, and an end (perhaps a holdover from the English/Arts backgrounds of many writers), to seeing documentation as structured components of information. You can then take these components and fit them together as needed, as if they were blocks of Lego.

Getting to the point where you can use XML effectively requires a lot of work – from learning the language to choosing an XML-enabled tool to analyzing documents to building DTDs. In the end, though, all the effort expended will be worthwhile. Moving to XML might not improve the quality of your documentation, but it can streamline the process of creating and managing documents.

Appendix 1: A Manual Chapter Coded in XML

This chapter from a manual for a fictitious XML editor illustrates one way to tag a document.

```
<chapter>
  <heading>
    <chapter.title>Using Your XML Editor</chapter.title>
  </heading>
  <chapter.body>
    <title>Using Your XML Editor</title>

    <section.overview>
      <section.title>Introduction</section.title>
      <para>Your XML editor is one of your most important tools.</para>
    </section.overview>

    <section>
      <section.title>What the Editor Can Do for You</section.title>
      <para>This XML editor has many features to assist you, including:</para>
      <bullet.list>
        <bulletlist.item>Easy editing in a standard environment</bulletlist.item>
```

```
<bulletlist.item>Automatic generation of a DTD</bulletlist.item>
</bullet.list>
</section>

<section>
<section.title>Starting the Editor</section.title>
<para>This section describes how to start your XML editor.</para>
<procedure.title>To start the editor:</procedure.title>
<procedure>
<procedure.item>On the desktop, find the icon
for the editor</procedure.item>
<procedure.item>Double click the icon.</procedure.item>
</procedure>
</section>
</chapter.body>
</chapter>
```

Appendix 2: XML Resources

The following printed and online resources offer you more information on XML.

Books

- *Mastering XML* by Linda Burman, Chuck White, and Anne Navarro. Published by SYBEX.
- *Learning XML* by Erik T. Ray. Published by O'Reilly & Associates.
- *XML: A Primer* by Simon St. Laurent. Published by MIS Press.
- *XML in Plain English* by Sandra E. Eddy. Published by M&T Books.
- *The XML Companion* by Neil Bradley. Published by Addison-Wesley.
- *Teach Yourself XML in 21 Days* by Simon North and Paul Hermans. Published by SAMS.
- *XML Pocket Reference* by Robert Eckstein. Published by O'Reilly & Associates.
- *DocBook: The Definitive Guide* by Norm Walsh and Leonard Mullener. Published by O'Reilly & Associates.

Web Sites

- DocBook Web site
- World Wide Web Consortium XML site
- Robin Cover's SGML/XML page
- XML.com
- XML 101
- XMLelephant
- FineTuning.com
- SGML/XML Web page
- webreference.com
- Web Developer's Virtual Library