



Australian Society for Technical Communication (NSW)

XML, why would you bother?

22 February 2006

Peter Meyer, Managing director, Elkera Pty Limited

Elkera Pty Limited ABN 68 092 447 428
Suite 701, 10 Help Street, Chatswood, NSW 2067, Australia
PO Box 5280, West Chatswood, NSW 1515
Telephone +61 2 8440 6999
Facsimile +61 2 8440 6988
www.elkera.com

Contents

Why are we talking about XML?	1
Some key needs in product documentation systems	1
Collaborative writing and quality control	1
Knowledge management	1
Output file formats	2
Market responsiveness	2
Content sharing and re-use	2
Integration with other production processes	2
Publication formatting	3
Product documentation begins with the specifications	3
Product life cycle management	3
Translation management	4
The problem with native Word documents	4
How does XML meet these needs?	5
The big question	5
The many languages of XML	5
XML in Word	6
The problem with presentation oriented schema	7
Structural schema	7
Some points about writing structured XML	8
What is unique about using an XML editor?	8

The problem with XML editors.....	9
Overcoming these problems.....	9
Conclusions.....	10

XML, why would you bother?

Why are we talking about XML?

This discussion is titled “XML, Why would you bother?” This is a rather loaded question. It assumes that people are thinking about XML but they may have some doubts.

Before taking on the question, it is necessary to provide some context. What are the problems that we are trying to solve when we think of XML? This will ensure that we can properly consider the merits or otherwise of XML in solving those problems.

Some key needs in product documentation systems

Collaborative writing and quality control

Publications may be produced by technical writers in different branches of an organization in different cities or countries. Writers need controlled, shared access to content components.

Parts of documentation may be written by software developers and other parts by specialist technical writers.

Contributions must be coordinated and changes approved by persons with relevant knowledge of publication issues.

Content must be maintained in a common format that can be output to meet local market requirements.

Knowledge management

Over time, personnel turnover will occur. Experienced writers may leave and new writers will be introduced to the project. It can't be assumed that writers know all existing content. In very large projects, even experienced team members may not know everything that is available.

Content must be exposed to writers in a way that they can easily find existing content dealing with a particular feature or process so as to not create unnecessary duplication or slow down production times. This requires rich metadata models that tie content to product components, features and user actions. This issue arises again in the section dealing with product life cycle management.

Output file formats

Product documentation may require these and other outputs:

- printed manuals and guides
- ready to print, electronic documents (PDF files for printed manuals)
- HTML
- Various flavours of Help, .hlp, .chm, Vista help etc
- Multi media training materials
- Outputs for mobile devices.

Often these outputs are quite distinct with little content that is common between, say print and HTML/Help but this is not always so.

Market responsiveness

Market pressures are promoting faster product development life cycles and, in the case of software development, new development tools are making this possible. Product release cannot be held up while documentation is prepared or updated. Thus production processes must be optimized so that only necessary content components are revised and reviewed. Production processes must be automated to minimize the need for manual intervention in processes such as formatting and hypertext link management.

Content sharing and re-use

In product families and even among different product families, documentation may include common corporate information and product information. It is costly and error prone to maintain this separately in each publication. It should be possible to manage content in components (topics) in one place and incorporate it into multiple publications.

Stored content components must be capable of being inserted at arbitrary points in document hierarchies and in multiple publications using quite different formatting.

This is not just a technological problem. It requires great care and skill in planning the information architecture and in the preparation of content.

Integration with other production processes

Some product information such as interface component descriptions and API descriptions may be written by software developers and may be created within the code files. Using a literate programming model, this documentation should be incorporated into the product documentation. However, it may require review by documentation

managers. To ensure consistency in the code base and the product documentation, two way linkages will be required.

When changes are made by one or other contributor, these may need to be flagged for review by the others.

Developers need access to the documentation for incorporation of links for context sensitise help in interface components.

Publication formatting

Publications may require quite different presentation in print, on the web or in other formats. At the very least, different fonts may be used but more extensive changes may occur. Contents listings, may be output quite differently. Hypertext links must be activated and validated in HTML and help outputs. Graphics must be rendered at appropriate resolutions for the output medium. Footnotes may be displayed quite differently in print and online versions. Print versions may be rendered in a single output but online versions chunked in separate files by topic.

Over time, software versions may be revised that require changes to proprietary file formats. Corporate styles may be revised to reflect new corporate images or changes following corporate mergers. It should not be necessary to edit content to accommodate any of these needs.

Product documentation begins with the specifications

Most products begin life in a vision statement, requirements document and detailed specification. If well written, particularly with use cases, this documentation may provide a substantial input into later product documentation. It may be valuable if specifications are managed in a compatible form with product documentation so that content can be incorporated without format transformation.

Product life cycle management

Increasingly, there is a need to more actively manage the total product life cycle to achieve quality outcomes. Product support and maintenance issues must be fed into the product development process and then into product documentation and support knowledge bases. It may be desirable to manage documentation in compatible formats at all stages in the product life cycle for consistency of metadata, content sharing and searching.

Translation management

Increasingly, companies must provide documentation in multiple languages. Translation costs are very high so it is necessary to manage content to minimize those costs. This may require the use of fine grained translation memory systems.

The problem with native Word documents

Word processor applications are designed to let authors create documents that can have almost any layout and style that might be needed in an office environment. Writers can choose single or multiple columns, apply paragraph and text style properties, create automatic numbering, create headers, footers, indexes, contents listings and cover pages to meet a vast range of document publishing needs. Applications such as Microsoft Word are commonly known as What You See Is What You Get (*WYSIWYG*) editors. What you see on screen ought to be the same or almost the same as what you see in print. We all know that this is not always the case, particularly when you send your Word document to someone else to print.

Word processing documents have three important characteristics:

- (a) They store information in a simple paragraph model in which each new paragraph is created when you press the Enter key. This means that there is no explicit relationship in the data to connect headings to the paragraphs to which they relate or introductory text to the following list items.
- (b) Word documents store format information with the content, even when named styles are used. This means that if you want to publish the document in another output, other than print, it is necessary to manually change style properties or apply new styles where the old styles don't match the new output.
- (c) It is difficult to store non printable metadata about particular components because the components may not be defined (a chapter or because there is no place to insert the information in a form that software can reliably process.

The effect of these characteristics is that if you want to publish a Word document to the web in HTML, the HTML document will look fairly similar to the print. If you want to change the fonts in particular places to improve the reader experience on the web or chunk a large document into single pages for each chapter or schedule, it is very likely this will not work as expected. Problems may include inappropriate or inconsistent formatting, poorly defined tables, incorrect chunking and broken or missing links to internal components and other documents.

If the document contains rigorously applied styles, it may be possible to get a better result. Unfortunately, most authors don't use Word styles. Most who do so change them arbitrarily, override them or apply local formatting instead. Style names may be difficult to interpret by anyone other than the author if they are based on their appearance, rather than the generic function of the content to which they are applied. Even with styles, the

absence of metadata can cause problems with links and non compliance with web accessibility guidelines.

It is possible to use Word documents for smaller documentation projects or those where many of the needs listed earlier are not of high importance. The basic problem with using Word for content writing is that the resulting system will not scale well to meet more of the listed needs. More and more effort will be devoted to manual processes and correcting errors.

How does XML meet these needs?

The big question

Before we can answer this question, we must understand some important points about XML. Since Microsoft is increasing the use of XML in Microsoft Word, we will also need to understand the implementation of XML in Word by Microsoft.

The many languages of XML

XML is not a markup language but a tool to define document or data description languages. Anyone can define an XML language to suit their particular needs. Thus, it is not helpful to say that something is “in XML” in order to convey important information. We must know which XML language is in use and its characteristics. The XML language in question may be useful or not for a particular purpose according to its design.

There are many commonly available XML document languages provided by standards bodies and developers. XML languages are defined in a Document Type Definition (*DTD*) or other schema definition language. These will be referred to as *schema*. The schema defines the grammar for the XML language. This includes the names of allowable elements and attributes, the order in which they may occur and other properties. A schema for document markup is a means of enforcing desired structure and business rules in content production. It ensures that data is predictable for use in automated processing systems. This is something that is not possible with word processing documents.

Some schema such as XHTML 1.0 (<http://www.w3.org/TR/xhtml1>) can be applied to a wide range of common documents as they may appear on a web page. However, such a schema defines only a very few generic structures in documents, such as heading, paragraph, list and blockquote. Often, formatting information is added to distinguish different kinds of information in the document. If you want to define a component of a document for special processing and make sure it is only used once and in a particular part of a document, you can't do this with XHTML 1.0. In addition, you cannot reliably determine the hierarchy of your documents. Headings (H1 to H6) can be used in any

order and at any level in the document. They are not tied to the paragraphs to which they relate.

Schema such as XHTML 1.0 are said to be flat, presentation oriented schema.

Other schema, including XHTML 2.0 (<http://www.w3.org/TR/xhtml2>), DocBook (<http://www.oasis-open.org/specs/index.php#dbv4.1>), DITA (<http://www.oasis-open.org/specs/index.php#ditav1.0>) and Elkerá BNML (http://www.elkera.com/cms/products/bnml_schema/) provide various ways to represent the true hierarchy of objects within the document. Some of these schema allow you to generically define particular parts of documents that may require particular processing in rendering applications or in content re-use or for searching. These schema describe the generic hierarchical structure of documents (chapters, parts, clauses, procedures, steps etc.). While each of the mentioned schema have this characteristic, they are very different and suited to different uses. A comparison of key features in the listed schema is available at http://www.elkera.com/cms/articles/technical_papers/comparison_of_xml_schema_for_narrative_documents/.

Many of the benefits from using XML can be obtained only if the application separates presentation information from the information itself. This allows software applications to read the XML file and select chosen components for particular kinds of processing that are needed for the information defined by that component. For example, a document abstract or synopsis may be suppressed in the print version but shown in a special location on a web version. Alternatively, different style properties may be applied in the print and web versions. XML schema achieve this by using names for elements that describe their function in all documents of a particular kind. In this way, the names are said to be generic. XHTML 2.0, DocBook, DITA and BNML are considered to be *generic structured schema*.

Generic structure schema can capture a lot of information about the content of a document, depending on the richness desired in the language. Such schema can provide great flexibility in the way document content can be searched, manipulated and rendered by software applications. This kind of flexibility is not available for documents using flat, presentation oriented schema such as XHTML 1.0 or the XML formats used in Microsoft Word.

XML in Word

Versions of Microsoft Word up to Office 2003 create a native or binary file format (.doc) by default when a document is saved. Word can also save out a text based representation called Rich Text Format (*RTF*). Recent versions of Word can also save HTML. From Office 2003, Word can optionally save WordProcessingML, Microsoft's own XML document format. Today, it appears that most users continue to save their documents as Word binary files (.doc).

From Microsoft Office 12 to be released in 2006, Microsoft advertises that Office applications, including Word, will save files in the Microsoft Office Open XML formats by default. From that point, there is no difference between a Word document and an XML document, except that all XML formats may not be equal. The new XML format for Word is an extension of the WordProcessingML format used in Office 2003. It is necessary to understand what kind of XML is produced by Word and assess it against your requirements.

In its preview materials for Office products *Microsoft Office Open XML Formats Frequently Asked Questions*

(<http://www.microsoft.com/office/preview/developers/filefaq.msp>), Microsoft makes it clear that its XML formats in Office 12 are “display oriented”, in the same way as WordProcessingML in Office 2003.

The problem with presentation oriented schema

Writers of documents in Word can create an XML document using all the same styles they use now. Using the default schema, Word does not impose any meaningful structure rules on content writers. There is nothing to tie headings to the content to which they relate and nothing to enforce a regular document hierarchy. Based on a formatting whim of the writer, a heading 1 can follow a heading 3 but semantically be part of the heading 3 topic, just as in a conventional word processor.

Presentation oriented schema are little different to common word processing documents in providing a foundation for automated processing. They do allow use of XML tools for processing but the semantic information in the markup often is missing or cannot be trusted, just as in traditional Word documents.

Structural schema

Structural schema such as DITA, DocBook, S1000D, Elkera BNML and similar schema impose rules on content writers that may require the use of elements in particular sequence and mandate the use of certain elements. These rules are enforced through validation of the document against the schema. Some or all these rules must be understood by the writer before he or she can effectively write content using an XML editor. This makes writing content with a structural schema different to using a word processor such as Word.

The key features of structural schema include:

- Information is self describing though the use of descriptive element names. Presentation information is separated from document structure so that any desired formatting can be applied automatically for each output. Content does not have to be revised to change styles or file formats in generated publications.

- Consistent document structure can be enforced by the schema. These characteristics remove ambiguity from processing systems and allow reliable automation that minimizes human intervention.
- Depending on the schema design, content components can be incorporated into multiple documents at arbitrary levels in the document hierarchy without re-tagging or re-formatting.
- Metadata can be attached to content components to assist information retrieval and processing. Structural containers ensure that metadata attaches to the right content. This facilitates reliable processing and searching.

It is clear from this short list that structured XML can provide the foundation for meeting most or all the needs listed earlier.

Some points about writing structured XML

What is unique about using an XML editor?

When using an XML editor, a writer must insert a valid element defined by the schema before typing content. This is quite different to a word processor where the writer can type, press Enter and continue typing to create new content.

After completing the content for an element in an XML editor, the writer has to:

- (1) identify and choose the next element they wish to insert either inside or after the current element;
- (2) if the element is outside the current element, find the valid location at which to insert the new element, usually by moving the insertion point;
- (3) insert the selected element;
- (4) if elements are nested, the writer may have to go back and pick another element to insert before writing new content; and
- (5) continue entering content.

In many XML editors, steps 1 and 2 are bound together. Commonly, the editor provides a pick list of valid elements at each location. The problem is that when the writer finishes a paragraph, the allowable elements that may follow that element cannot be seen until the insertion point is moved to the correct location. This can be quite frustrating, particularly for writers who are new to XML editors.

Other demands imposed on writers in XML editors include:

- The schema may require an element to be inserted or an attribute value to be completed. If it is omitted, the document is invalid.

- When an element has to be moved, it can be moved only to a valid location. If the writer does not know how to find the valid location, moving content can be very frustrating.

Many XML editors provide multiple views of the document, including normal or Tags Off View, Tags On View and a Structure View. To deal with the problems mentioned, writers often have to work in a Tags On or Structured View.

The problem with XML editors

The particular characteristics of XML editors impose several burdens that are not as pronounced when using a word processor. This leads some organizations to try to let technical writers work in Word using style templates. Documents may be converted to XML later in the work flow. This is rarely entirely satisfactory. It is difficult, if not impossible, to reliably convert word processor documents to structural XML without some human intervention to correct errors.

Writers using an XML editor may need a good understanding of the schema rules to understand how to find valid locations for new and moved elements. This imposes a high training burden and may lead some writers to be intimidated by XML editors at first.

The additional steps introduced into the writing process by XML editors can slow down work and break concentration while writing.

If the use of XML editors is to become more widespread, it will be necessary to simplify the process for writers and enable them to work more easily.

Overcoming these problems

Schema design

All the problems with XML editors can be overcome through good schema design and thoughtful customization of the editor to the schema.

Ideally, the schema should be the simplest model that will create the desired markup structure. Most documents conform to very standard patterns for section/clause, paragraph and list structures. Ideally, the schema should make it easy for the writer to understand how these work. If the interface is then designed carefully, the writer won't really need to know the finer points of the schema rules until they are ready to learn them.

Editor customization

Most XML editors are designed to be customized in some way to provide writers with various shortcuts and aids to their work. XML editors must work with a wide range of schema. It is not practicable to treat them as a ready-to-use product. Sometimes,

extensive customization is required to achieve a highly usable writing interface. The capacity of an XML editor to facilitate that customization should be carefully considered during evaluation.

The key to customizing an XML editor is to fully understand the content to be created and the conventions followed by writers. A customized interface need not try to deal with every situation. It should aim to make work easier for that common actions that will likely amount to 80% of the writer's work.

A good test of an effective interface is if the writer can perform most work without seeing any tags. While some editors attempt to work wholly without tags, Tags On Views can be very helpful to enable writers to quickly and accurately perform unusual editing actions.

When properly implemented, writers should find that content writing in XML is much simpler than using a word processor. The freedom to forget about presentation and concentrate solely on structure is liberating. Writers no longer have to concern themselves whether automatic numbering and cross references will break as content is edited or how various fiddly layouts will be created. All these processes can be handled automatically.

Conclusions

The choice between using Word, including Word XML or using structured XML involves a clear trade-off.

The use of Word avoids a layer of change management within the organization. Most users already have Word on their desktops. The temptation is high to use it if you can. Senior managers who don't understand document publishing issues rarely understand why Word is not good enough. This makes it hard to obtain funding for new software. It seems cheaper up front.

The costs of using Word or format oriented XML may not show up immediately but they can be substantial and occur over a long period. They may include:

- low functionality in basic systems with poor scalability to meet more complex needs as the business grows;
- costly manual re-purposing of data to manage different outputs and hypertext linking or simply not being able to meet real customer needs in documentation;
- excessive duplication of content resulting in inconsistencies, errors and higher production and maintenance costs;
- slower production cycles with impacts on time to market for new and updated products.

Put simply, Word was never designed to produce content for use in automated document publishing systems such as those needed for complex product documentation.

The switch to structured XML involves a higher level of change management at the outset. In a few cases, there is a genuine fear of this change based on examples of poorly implemented XML systems which did not take sufficient account of the needs of writers. In other cases, it is based on no more than an assumption that “We must be able to do it in Word”. Structured XML is designed to enable systems to meet the listed needs. It will maximize functionality, reliability and scalability, thus reducing costs over the medium to longer term.

It is amazing that many organizations are making decisions affecting fundamental issues of content management system design based on little more than a poorly defined resistance to using a new writing tool.

In your enterprise, do you want to just keep using a word processor for its own sake or do you want to establish a framework that is designed to do the job at hand?