



Elkera Pty Limited

Introduction to single source publishing

20 March 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

The schism in enterprise content	1
An alternative approach.....	2
Why XML?	3
The limitations of word processing documents	3
The costs of using the wrong tool.....	4
Can Word can now create XML?.....	5
The many languages of XML.....	5
XML in Word.....	6
The problem with presentation oriented schema	7
Structural schema	7
Conclusions.....	8

Introduction to single source publishing

Many enterprises need to produce publications for users who will access information via print, web and possibly other mediums such as mobile devices. Content users may be insiders who will access information via an intranet or they may be outside customers who will access it via an internet site or by acquiring printed publications.

The publication of information in different mediums often requires quite different handling and presentation of content to take advantage of the characteristics of each medium and the needs of users. For example, it is rarely satisfactory to publish a facsimile of a printed document on the web. The web site version may require linked contents listings, active hypertext links between content chunks and other documents, re-ordering of some content, different resolutions and formats for graphics, screen friendly fonts and other features to meet accessibility requirements. Using a single data source to produce these outputs is called *single source publishing*.

The benefits from an effective single source publishing strategy can be considerable. Content production costs can be reduced and publishing cycles can be shortened. Documents can be published consistently in accordance with corporate styles and branding without having to edit content when styles change. Service delivery to content users can be greatly improved. When the right information is put into the hands of users in the right form, errors and support costs can be greatly reduced. A single source publishing model is all about putting the content user first and reaping the benefits.

This document explains the concept of single source publishing and discusses the core issues in content writing and management for single source publishing.

The schism in enterprise content

Most enterprise content is prepared using word processing software such as Microsoft Word. Increasingly, enterprise web sites and intranets are managed via content management systems. While called a “Content Management System” (CMS), most such systems are really intended as web publishing platforms that are concerned with handling content that is to be published on the web and with related web site functions.

Most web CMS systems require that content is prepared as HTML or XHTML using a dedicated “Rich Text Editor” or is published as a PDF file generated from word processing documents. The consequence of this is that on very many enterprise web sites and intranets, only basic site navigation and highly presentation oriented content is presented in HTML pages. However, substantive articles, reports and manuals commonly are available only in PDF documents. If HTML is provided, there is a good chance it will be a facsimile of the print or it will be affected by various formatting problems. The result is that content is not in the most convenient form for users.

Important information is not passed on because it is too hard to find or users waste time looking.

There are many software tools that attempt to convert Word documents to HTML and other formats. These conversions are rarely perfect and almost always involve significant effort and compromises, for reasons explained in more detail later. Word processing documents are not suited to reliable automation of publishing processes that involve presentation of information in different formats and layouts. If it were otherwise, we would not see the split between PDF and HTML on most web sites and intranets.

It is worth exploring why this schism occurs in content management, despite these obvious problems.

Notwithstanding these problems, there appears to be a tendency in many enterprises to insist that content must be created in Word without really considering whether it is in fact suitable for the task at hand. Remarkably, the choice of a particular tool is driving content management strategies when conventional wisdom would dictate that a tool should be chosen only if it is suited to the job at hand. There is an aphorism; *If the only tool we know is a hammer, everything looks like a nail*. In the content management area this might be translated to; *If the only tool we know is a word processor, everything looks like a print document*.

There are many reasons why this situation has developed. Business managers may not be aware of alternative strategies or of the true costs of the status quo, word processing based content management strategies. This reflects poor marketing by vendors of alternative strategies. Unfortunately, many of the tools advanced for true single source publishing have not meet satisfactory usability levels and they have been expensive to acquire and develop. Perhaps it is little wonder they have not made it on to management's radar screen outside of larger enterprises that produce high volumes of content that must be published in multiple outputs. That situation is now changing.

An alternative approach

There is an alternative approach to content management that will allow almost any enterprise to have the best of both worlds. It requires that relevant content is prepared and maintained in a way that permits it to be reliably and automatically output to any desired publishing format, whether print, RTF/Word, PDF, HTML, Help or anything else. An effective single source publishing strategy will also permit improved content sharing among multiple publications and more flexible creation of publications to suit particular target audiences.

As explained later in the section *Why XML?*, effective automation and control over output formats requires the use of structured markup for content. This cannot be achieved with standard word processing software. The most effective way to create structured content is to use a structured XML editor with an XML DTD or schema that

defines a suitable language to describe the content. XML DTDs and schema are explained in later sections.

To create structured content, writers of high value content must be asked set aside their word processing software and use a new writing tool. Whether this is practicable or a good idea will depend on several factors. These include the nature of the content to be produced, whether writers can be easily trained and supported, the usability of the structured XML authoring tool and the costs of change. It is not suggested that there is any need to change the tools used by writers of everyday office documents.

The decision to change content authoring software is one of the foundation issues in planning a single source publishing content management strategy. Do you insist on using Word as the content editor and accept the compromises that follow or do you introduce a new content writing tool that will enable greater automation, flexibility and reliability in content management and publishing processes?

Why XML?

The limitations of word processing 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 a word processing document to someone else to print.

Word processing documents have three important characteristics:

- 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.
- Word processing documents store format information with the content, even when named styles are used. When publishing the document in another output, other than print, it is usually necessary to manually change style properties or apply new styles where the old styles don't match the new output. The ability to automatically apply different formatting properties to styled paragraphs is limited by the amount of information conveyed by the style name and the consistency and accuracy of use of the style.
- It is difficult to store non printable metadata about particular components because the components may not be defined. Metadata cannot be added to a

chapter or clause because there is no place to insert the information in a form that software can reliably associate with all the content that belongs to those components.

The effect of these characteristics is that when a word processing document is published to the web in HTML, the HTML document will look fairly similar to the print. Attempts to completely re-style the content for the web such as 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 will almost always run into inconsistencies in the data or a lack of information. 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 named styles, it may be possible to get a better result. Unfortunately, most word processor authors don't use styles. Many who do so use them incorrectly, change them arbitrarily and override them with local formatting. 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.

The costs of using the wrong tool

It is possible to use word processing documents for smaller documentation projects or those where many of the needs listed earlier are not of high importance. The basic problem with using a word processor 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.

The costs of using a word processor in a single source publishing system 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 delays in the provision of updated information.

Can Word can now create XML?

Yes, recent versions of Word can create XML. Before looking at the XML created by Word, it is necessary to understand a little more about XML generally.

The many languages of XML

XML (<http://www.w3.org/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” unless we 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 document components. 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 Elkera 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 in the article Comparison of XML schema for narrative documents (http://www.elkera.com/cms/articles/technical_papers/comparison_of_xml_schema_for_narrative_documents/) by the author and Elkera senior consultant Andrew Squire.

Many of the benefits from using XML can be obtained only if the application separates presentation information from content. This is only possible if the role of each content element is described in a way that software rules can be written to reliably apply desired formatting in each published output. 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 structural schema*.

Generic structural 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).

In the forthcoming Microsoft Office 2007, 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. However, it is necessary to remember from the earlier discussion 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.mspx>), Microsoft makes it clear that its XML formats in Office 12 are “display oriented”, in the same way as WordProcessingML in Office 2003.

It is possible to use custom XML schema in Word in Office 2003 and Office 2007. Before choosing to use this approach, the usability of Word as a structured XML editor should be carefully assessed and compared with alternative tools.

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 XML schema such as Elkera BNML, DITA, DocBook, XHTML 2.0 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.

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.
- Most document content is organized in a hierarchical structure of topics, paragraphs and lists etc. Structural schema represent this hierarchy so that content can be reliably processed in logical units.
- 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.

The structural schema mentioned earlier are each very different and designed for different purposes. Care must be taken to match the right schema to your content needs. Often it will be necessary to customize an off-the-shelf schema. In some cases it may be necessary to develop a new schema.

Conclusions

The use of structured XML converts document “blobs” into smart resources that allow information to be reliably managed at a much finer level of granularity.

The separation of high value content from presentation formats and from proprietary processing applications ensures that production and publishing processes can be reliably automated. A single document source can be used in automated systems to produce many different outputs with quite different components and layouts. Content can be delivered in the most appropriate form to meet user needs.

Content delivery options should not be constrained by insistence on using a particular kind of a content writing tool. A range of satisfactory tools are available to create structured content. Content management strategies should be based on defined needs and the selection of tools that will meet those needs.