

XML Solves Publishing Problems

By: Sean Wheller
sean@inwords.co.za

If you are creating content or manage documents you have probably encountered or been faced with solving problems such as:

- single-sourcing
- collaborative authoring
- cross-platform editing
- multi-channel publishing
- improving information quality and consistency
- enhancing functionality of electronic output
- negating technology lock-in
- and even reducing costs without reducing team head count.

This paper explores how the use of XML technologies within your authoring system can help you achieve each of these objectives.

Table of Contents

| | |
|---|----|
| Introduction | 4 |
| From WYSIWYG to WYSIOO | 4 |
| Solving the Problems | 5 |
| Single-sourcing | 6 |
| Collaborative authoring | 6 |
| Cross-platform editing | 7 |
| Multi-channel publishing | 7 |
| Improving information quality and consistency | 8 |
| Enhancing functionality of electronic output..... | 8 |
| Negating vendor lock-in | 9 |
| Reducing costs | 9 |
| Where Next?..... | 10 |
| Conclusion..... | 10 |
| Exclusive Offer for “123 Gift for Me” | 11 |

Introduction

Technology changes the face of the future by solving past problems and opening avenues for new possibilities. So what happens when technology comes along that changes virtually every other technology? That's what we are about to find out as we explore the impact Extensible Markup Language (XML) is having on the publishing industry.

XML is a recommendation from the World Wide Web Consortium (W3C). Like HTML (Hyper Text Markup Language), XML is a markup language. Both have their roots in Standard Generalized Markup Language (SGML), but were designed with different goals in mind:

- XML was designed to describe data and to focus on what data is.
- HTML was designed to display data and to focus on how data looks.

XML therefore is not a replacement for HTML. HTML is purely about displaying information, while XML is solely about describing it. Both contain data, but XML separates the data from the presentation layer.

What does this mean to publishing departments? How does it benefit and change work processes for the better?

From WYSIWYG to WYSIOO

To understand the answers we must understand the problems publishing departments have with current technologies. However, before we do this we must understand a fundamental shift between the way authors currently interact with text using traditional technologies and how they will do so with the XML based technologies of the future.

A key difference between current technologies and XML based systems is the shift away from a WYSIWYG editing environment to WYSIOO.

WYSIWYG stands for "**What You See Is What You Get.**" WYSIOO stands for "**What You See Is One Option.**" The difference between these editing environments is founded in the different goals for which HTML and XML were designed.

Traditional publishing and help authoring tools provide a WYSIWYG environment for authors to compose their texts. They display pages of information on screen using layout and formatting that is a near exact rendition of how the page will look in the final product. In addition to enabling text composition, they also enable formatting and layout. Much the same as the HTML design goal.

This approach is not possible with XML since it was designed to describe data and to focus on what data is, not how data will be displayed or formatted. Authors composing texts stored in XML must use a "Structured Editor." This means the editor is focused on two tasks: text composition and valid markup thereof. Since formatting information cannot be saved inside the XML file, the best an editor can do is

render temporary formatting and layout for the duration of the editing session. What authors see during the editing session is "**One Option**" of what they may get in the final product, WYSIOO.

WYSIOO and WYSIWYG are opposite ends of the same stick. The two end-points cannot be brought together without breaking the stick. Try as you may, if you do manage to bring these polarities together you will find that you have violated the fundamental laws of one or both of these paradigms.

Storing information in a format that is presentation neutral and editing in an environment that does not perpetually mirror the layout and formatting of the final product is strange and disconcerting to people accustomed to working with WYSIWYG . The reason is that they have to relinquish control over the presentation layer. While this should be good news, as it relieves authors from having to pay attention to formatting and layout, it is often most disconcerting to people who are accustomed to using fonts, adding bold, italics and underscore to text. They are also very accustomed to viewing page layout and content flow in the comfort of knowing that, what they see on screen is how it will look in the output. Not being able to see life in this manner unnerves some authors to the extent that they find it hard to concentrate. As trivial as it may seem, some never get used to WYSIOO and prefer to remain with their current Word Processor and Help Authoring Tool.

However, separation of data and presentation layers is the key that enables publications departments to overcome problems with current technologies and employ new techniques that are better suited to the requirements of today and the future.

Solving the Problems

So what are the problems current tools cannot solve? What techniques do they not support and so fail in meeting objectives?

Broadly speaking there are seven areas where tech publications departments have problems that are hard to solve using traditional technologies, they include:

1. Single-sourcing
2. Collaborative authoring
3. Cross-platform editing
4. Multi-channel publishing
5. Content reuse
6. Enhance functionality
7. Negating vendor lock-in

The problems found in each of these areas hinder performance, quality, and cost improvements. Solutions to these problems, using traditional technologies, have been at best partial. They are generally cumbersome and require a high degree of user intervention. These solutions tended to solve a particular problem of one area, instead of addressing all areas in a manner that is cost effective, efficient and responsive.

Single-sourcing

The need to publish content to multiple platforms, user agents and user types has, more often than not, meant that a single document will be maintained for each target. This approach is fraught with problems, especially when it comes to maintaining a copy for print and online help outputs.

There has long been a latent need for technology that could maintain documents in a format suitable for conversion to all targets without incurring a large maintenance overhead. This gave rise to a concept called "single-sourcing." Long the "Holy Grail" of publishing, the idea is to write once and publish to many target formats from a single source, while the source format never being used for presentation, only for editing.

Tools such as RoboHelp, ForeHelp, and HDK all purported to be single-source tools that would enable a person to write once and publish to many. This was not really true, since each of these tools used the online help source format, a presentation format, as the master for authoring and therefore did a pretty poor job of converting to Word document or PDF.

With XML the approach is different. Documents are authored in XML, which, as we explained, is presentation neutral. The XML is then transformed to a target presentation format using Extensible Stylesheet Language (XSL), another XML format, and an XSL processor; an application that takes XML and XSL as inputs and outputs a new file. This file may be another XML file, a presentational format such as HTML, or XSL-FO a transitional format that can be transform to binary presentation files such as RTF, PDF, or PostScript.

This approach is far more effective since the transformation can include any number of custom parameters and functions that automate the process. The transformation from XML to any target format requires far fewer steps and requires minimal user intervention.

Collaborative authoring

In every authoring process there comes a point where input is required from other people. The problem with traditional formats is that they do not easily enable two or more people to work on the same document simultaneously. In times of pressure or short deadlines this can be a real problem.

What is needed is a way to allow people to work on the document in tandem, without "clobbering" each other's work or corrupting the file. XML alone does not solve this problem. However, since XML is a plain text format, as apposed to a binary format typically used by traditional tools, it is light weight, easy to transport and perfect for management under revision management systems such as Concurrent Version System (CVS) or Subversion (SVN). This means that all authors can obtain a copy or version of a document, simultaneously edit it and commit *only their changes* back to the revision system. Everyone works against a centralized, revision controlled repository. Updating their working copies and committing their changes back for everyone else to see. Note that we said, "*only their changes*" as apposed to saving the entire file back to the repository, which is what you literally do when using binary formats even when working under revision control.

In practice this method of managing documents is akin to the method used by multiple programmers who need to collaborate on the same code. In addition to the ability to work on the same document under revision management, managing XML documents in revision management systems such as CVS or SVN enables authors in any geographic location to use the Internet to collaborate on a project wherever they are.

Cross-platform editing

There are very few tools that enable technical authors to use any operating system or editor to open and edit all of their documents. In most cases, one is forced to use the operating system and a copy of the tool that was originally used to create the document. This is most inconvenient as writers are often required to document software that runs only on a Solaris, Linux, or OS/2 system, when their workstation has a variant of Windows installed and a Windows based Word Processor such as Word .

To get around the problem an author has to compose on one computer and experiment on another. This is less than efficient since it means that additional computers are required for each writer, or a single computer must be 'time-shared.' It also slows down the authoring process as the author is frequently forced to move between machines. It is far more efficient to provide an editor that can run on all platforms. One solution may be Open Office, but this would mean installing a large application for what is essentially a simple task. While this is a nice to have situation, with XML it is not mandatory that the application first used to create a document be installed on each computer where the document will be edited. This has the nice side-effect of enabling authors to choose to use any combination of their favorite tools without impacting on, or creating a situation where other authors must use the same tools.

Just as XML is presentation-neutral, it is also platform-neutral. Meaning that it can be opened and edited on any OS using any XML or plain text editor. So you can create the document on Windows using a structured editor and then use a plain text editor on any other platform, many of which have tools such as VI or emacs pre-installed. The tools provide modules for structured editing.

This last point is hugely advantageous in environments where people need to share documents with colleagues or partners located on the other side of the globe and have no influence on the tools they may use or their organization provides.

Multi-channel publishing

The concept of multi-channel publishing extends the concept of single-sourcing to include the ability to maintain a single source containing content for all platform, or customer, or user specific differences and nuances. In other words, several types of document can be derived from the same source. For example: your product runs on Linux and Windows. Each version provides OS specific options. You want to capture screens with the look and feel of the OS and document these options. Since the options are OS specific it would be redundant to put them in both books.

Using XML it is possible to use a technique called 'profiling.' OS specific screens and options are specially tagged so that they can be included or excluded from the output. Profiling makes it possible for technical authors to author in a single place, making optimum use of common texts and profiling those texts that are OS, customer, or user specific. This saves time, reduces content duplication and delivers information into the output that is relevant to the version and environment the user is working with.

Improving information quality and consistency

Improving quality is a responsibility shared by the entire organization. An XML system, employing the techniques described, goes a long way to enabling the whole organization to collaborate on and contribute to quality. The portability of XML documents enables other departments to open and edit using any text editor as and when they spot a problem. These changes are tracked by the revision management system where they can be spotted by the authoring group and edited for consistency. The combined and distributed effort of many will mean that documents are kept current with less effort. Authors spend less time doing errata and can focus on adding new sections.

With an XML based system the improvement in quality and consistency is also achieved in other ways. Authors have no control over formatting while writing. This is done at the end of the writing process by the stylesheets. They determine all aspects of layout and formatting. Since it is not possible to obtain a formatted output without transforming the XML document using XSL, locating stylesheets in a central location and limiting their modification to a few people, ensures that consistency of layout and formatting can be governed across the organization.

Another technique made easily possible is 'modular content creation.' The structured format of XML makes it possible to break large chunks of text into smaller parts that can be assembled into multiple larger documents. This significantly increases 'content reuse.' The idea is to write a small chunk of information into a single file and include the text at the appropriate location by reference. The information is not duplicated but stored once and dynamically inserted into the main document during transformation. Increase in content reuse has, over time, an exponential impact on reducing the time required to update information. An update made to a single file is inherited by many larger documents and automatically reflected within the next transformation.

Once again, this concept is akin to the way programmers write Object Orientated (OO) code. Already existing objects are reused many times. Modular documentation and content reuse techniques serve to shave thousands of dollars off development costs when implemented correctly.

Enhancing functionality of electronic output

The engines or viewers used to open and render traditional electronic formats have been limited to a specific set of functionality. This made it difficult for developers to add features and functionality to the document. For example: Let's assume you have a need for users to be able to add comments in line to the paragraphs on a help topic. This would not be possible with traditional help authoring solutions. With XML transformed to HTML, combined with JavaScript and running under a Web Browser this is now possible.

The ability to achieve this objective is made possible, once again, because the transformation scenario can include any number of parameters or functions. By programming the transformation to include the specific scripts and integrate them with the transformed presentation output the required functionality is achieved. This step need only be done once and reused countless times by everyone in the organization with the exact same results in formatting and functionality.

Negating vendor lock-in

This objective is concerned with ensuring that the authoring system is free from proprietary solutions that may become obsolete or tie the organization into a specific vendor product, which has traditionally been the case. Departments based on Microsoft Word or RoboHelp have found that these tools have locked them in to their use for the foreseeable future and excluded them from solving the problems already mentioned.

With XML the risk of technology lock-in by proprietary tools is negated since any text editor can open and edit an XML file. The W3C standards ensure this interoperability.

An example that is soon to play itself out is the new online help format that Microsoft will ship with its' new product. Code named "Vista," the help engine is based on XML and is a proprietary extension to the Open Source format known as Docbook.

The problem is that publications departments using traditional tools will have to convert all their existing documents to the Vista format. No doubt vendors will provide upgrade products that port to Vista, but the department will still have a cost in licensing and training. These costs would be avoided if they had been working in XML since authors would be focused on content and not formatting or layout. A new set of stylesheets would take care of Vista requirements on their behalf.

This example demonstrates how using an XML based authoring system can prevent vendor lock-in and ensure compatibility with future technologies and the requirements they bring.

Reducing costs

Cost reduction is a major initiative of most organizations today. Everybody wants to do more with less. Cost reduction often results in team reduction. This places additional pressure on remaining team members.

By employing XML and the practices outlined in this document, publication departments can realize significant cost reductions, without losing team members or sacrificing on improvements in quality and performance. However, before making the change one must first evaluate the technology to determine whether or not XML is a suitable solution.

Do you have a need for the objectives we have discussed? If not, then XML is very likely not a solution for you. The cost incurred in moving to such a system could not be justified. However, just because you could benefit from a few of these benefits, also does not justify a move without a solid business case.

In addition, it is wise to keep in mind that the decision to implement such a system is a strategic one. Return On Investment (ROI), is not immediate and the Total Cost of Ownership (TOC), is not always lower than the current system in every instance.

Now building a business case is not an easy task. Most people would be seriously challenged if asked to build a business case. Fortunately an example business case is available at IDEAlliance . The document, entitled "Developing a Business Case for an XML Authoring System was written by Jean Mercedes Hamilton, a Senior IT Project Manager at SPX Valley Forge, Munich, Germany. It provides a template on which anyone can build a business case that will knock the managers socks off and curl their toes into a fist.

Where Next?

There are many who would recommend that the best way to implement an XML authoring system is to write your own XML Document Type Definition (DTD) and XSL stylesheets. Of course this is the best option since you can custom design the system to meet your exacting requirements, but this approach requires time and money; both commodities are normally in short supply.

To avoid this problem others and myself recommend adopting an already established, mature and Open Source application already mentioned in this document, Docbook. Docbook is a standard managed by OASIS. Docbook defines a markup vocabulary that is particularly well suited to describing technical literature. Managed as an Open Source Project at SourceForge, Docbook is the standard for storage of documentation on Open Source projects, but has also been widely adopted by commercial projects. This standard is by far the most popular starting point for organizations that want to get started with the implementation of an XML authoring system. It provides a comprehensive DTD that covers practically every tag you will ever need to describe in documents and books. The project also provides a robust set of XSL stylesheets that enable transformation to formats such as Rich Text Format (RTF), Portable Document Format (PDF), HTML/XHTML, HTML Help, JavaHelp and PostScript.

Docbook has a large Open Source following. The community not only develops the application but also supports it via its' mailing list. In addition there are two books written on Docbook that can help authors get started and productive. Cost and timesavings achieved by using Docbook are immense when compared with the cost and time for rolling your own.

Conclusion

Changes in technology and the market environment have brought about new requirements that traditional tools and technologies cannot satisfy. An authoring system based on XML technologies enables organizations to meet these requirements while increasing their level of execution and reducing development costs. However, XML authoring systems are not applicable to every situation.

Organizations must first evaluate the technology to determine whether or not the initial investment in such a system will add value. Providing a positive business case can be made, there is no need to build a custom system from the ground up. Open Source technologies based on the Docbook standard provide a cost effective way to implement a system and realize a ROI in a shorter period. The decision to move an authoring system to XML is a strategic one. While any such initiative should be founded on a solid business case, numbers alone are not the only reason for implementation. Factors such as technology lock-in and future compatibility can help organizations to avoid future surprises and unexpected costs.

Exclusive Offer for “123 Gift for Me”

FREE 1 Hour Consultation on CMS Websites

Receive a DEMO on how to create superior websites for FREE, with easy-to-use content management tools and easily maintainable by anyone
(and the best part you don't need to be a programmer or even know HTML.)

**** (This consultation is curtesy of our sister company TheWebDesignCompany.co.za) ****

This FREE 1 hour consultation is valid only for registrations entered in before the **22 of December.**

After the 22nd of December, we will charge regular price on all consultations.

We will only consult people that we TRULY believe we can HELP solve their publishing problems and add VALUE to THEIR BUSINESS.

ONLY fully completed registrations will be considered for consultation. This is highly important, as we need to allocate the appropriate consultants with the correct solutions. Therefore, please fill in the applications **FULLY** and as **ACCURATELY** as you possibly can.

After the allocated registrations we will continue to give valuable information via our newsletter, so don't forget to subscribe to our website. Our newsletter will keep you informed of all the advanced publishing and writing solutions.

To receive your GIFT of one hour FREE CONSULTATION please send email to: 123GFM@inwords.co.za containing the following details:

1. Subject line: Websites 123GFM
2. Full Name of contact person
3. Skype username [all consultations are done via skype (www.skype.com)], unless you prefer to contact us via landline. Please state your preference.
4. Industry business operates in (e.g. automotive, health, internet, software, publishing, etc)
5. What problem would you like to resolve in this one-hour consultation? Please provide description of problem and background, also some background about the person receiving the consultation.

✧ We will confirm your FREE Consultation via email to arrange a date and time. ✧