Editing within and for markup languages

Essential skills for modern editors

Paper presented at the IPEd national conference
Write | Edit | Index, 9 May 2015, Canberra

by Linda Nix
Golden Orb Creative
www.goldenorbcreative.com

Dr Linda Nix, BA Hons (English) PhD (History) Grad Dip Computing, is a professional editor with 20 years of industry experience in editing and production for print and digital formats, including hands-on experience with a range of markup languages and schemas. Her editing expertise includes business, law, finance and accounting publications, trade non-fiction and literary fiction. She has run her freelance business Golden Orb Creative since mid-2010 and publishing imprint Lacuna since late 2012.

Why talk about markup languages?

For editors working in professional publishing, there is no escape from markup languages. An article reviewing the use and range of markup languages over the last 15 years interviewed a number of publishers and publishing technology providers, all agreed on the importance of markup languages in 2015:

> no matter where the markup language is implemented in a publisher's workflow, it is definitely implemented
>
> Marianne Calilhanna, Cenveo Publisher Services,
> in *Publishers Weekly*, 24 April 2015

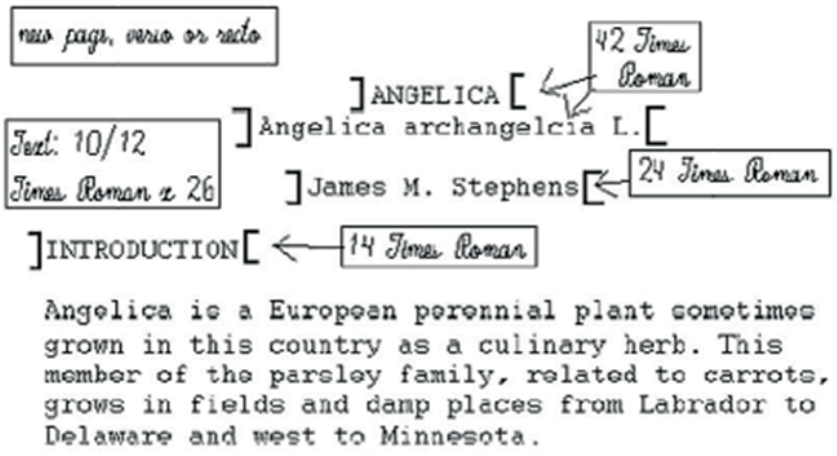
But there is a perception among publishers, and sometimes among editors themselves, that the technology is too complicated for editors, that editors are technophobes.

This paper argues that editors who are daunted by the alphabet soup of markup languages – XML, XHTML, HTML, SGML – are more than capable of understanding the principles of markup.

What do we mean by markup?

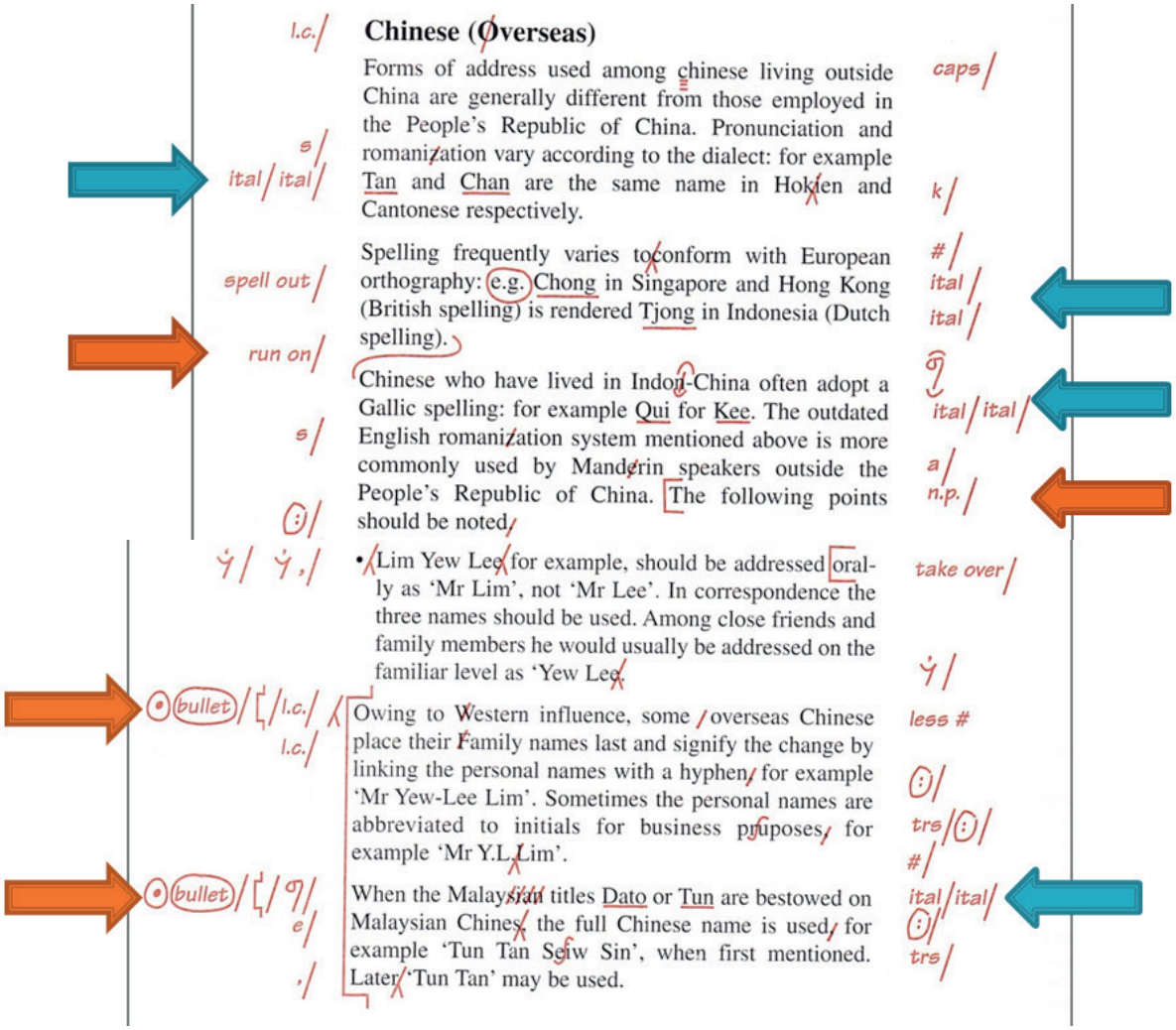
There are three kinds of markup.

1. **Presentation** markup describes how data should be displayed (font, size, colour etc). On paper this is shown as:



2. **Structural** markup describes data in:
   - general structural terms, eg headings, paragraphs, quotes, lists
   - specific semantic terms, ie what kind of data it is: title, author name, abstract heading, abstract, body heading, body paragraph, footnote, reference.
3. **Corrections** markup describes changes to be made to data, and is familiar to editors who use either standard proofreading symbols on paper or Track Changes in MS Word to show corrections.

The example below, taken from the *Style Guide* 6th edn, shows a combination of all three kinds of markup applied to hard copy. Presentation markup is indicated by the blue arrows, where italic style is to be applied, while structural markup is shown with the orange arrows, where paragraphs and bullets are defined.



Markup languages (MLs)

Digital markup languages are defined in standards specified by the World Wide Web Consortium (W3C.org). The main ones in use are:

- **HTML**: HyperText Markup Language – describes *presentation* (formatting).
- **XML**: eXtensible Markup Language – describes *structure*.
- **XHTML**: a subset of HTML structured to be compatible with XML or, in other words, an XML version of HTML.

HTML and XHTML can be used with another W3C standard: **CSS** (**C**ascading **S**tyle **S**heet) which describes the formatting to be applied to the structure defined in the XHTML or HTML.

XHTML and HTML are used for presenting text on web browsers, whether it appears on a website or an application that uses web browser technology such as intranets, ebooks and some apps.

XML is almost limitless in application and frequently defines the context of XHTML documents.

XML for data transfer

The W3C defines XML as "a software- and hardware-independent tool for *carrying* information", noting that "XML does not 'do' anything".

This is because XML files are **plain text** files that are:

- readable on any machine (hardware)
- readable by any app (software).

Other applications determine how XML coded data is read, stored, used and displayed.

For these reasons, XML is de facto standard for data exchange on the internet.

XML is everywhere

XML is used throughout the publishing industry to structure and transmit data. Here are the main examples of XML standards in everyday use.

Scientific publishing XML schemas

- **JATS** (formerly NLM) for archiving and publishing scientific, technical and medical (STM) journal articles.
- **BITS** for STM books.
- **MathML** for mathematics. There are two versions of MathML, a presentation version to describe how an equation should be displayed, and a semantic version to describe the operation of an equation.

Book publishing XML schemas

- **EPUB** for reflowable and fixed layout ebooks. EPUBs are packages that include both XML and XHTML files, as well as other kinds of files.
- **DAISY** for talking books for visually-impaired readers.
- **ONIX** for book metadata.

Technical and reference publishing XML

- **DITA** for anything topic-based.

Microsoft Word XML

Since 2007, Microsoft Word uses its own XML schema – the 'x' in .docx stands for XML. Every editor who uses Word is using XML, even if they don't know it.

HTML is everywhere too

As mentioned, HTML is the standard for web browser display and is used in browser apps, namely Chrome, Firefox, Safari, Internet Explorer; ebook apps, eg Bluefire, iBooks, Kindle, Kobo, Nook; and browser-based interfaces to other systems such as databases and content management systems (CMS), eg website and blog software.

XML and XHTML tags

Structural markup is applied through "tags". XML tags are semantic, ie they have meaning and are readily understood by English-speaking humans. XHTML tags are structural and look more like code than English, but they are very limited in number and easy to learn. Tags are indicated by angled brackets. Following are some examples of both XML and XHTML tags.

```
<title>This is an XML tag for a title element.</title>

<title date="08052015">This is an XML tag for the title
element with an attribute "date" containing the value of
today's date in DDMMYYYY format.</title>

<h1>This is a top heading level tag in XHTML.</h1>

<h1 class="title">This is the same XHTML heading tag with
the class "title".</h1>
```

Example

In the example below, the same text is given an XML tag in the first column, and an HTML tag in the third column. Thus the title of the document is tagged semantically as `<title>` in XML and structurally as the first level heading `<h1>` in XHTML.

XML	TEXT	HTML
title	Wuthering Heights	h1
author	Emily Brontë	h2
chapter	Chapter 1	h3 (or h1)
para 1 year	**1801**.—I have just returned from a visit to my landlord—the solitary neighbour that I shall be troubled with. This is certainly a beautiful country! In all England, I do not believe that I could have fixed on a situation so completely removed from the stir of society. …	p strong
para	'Mr. Heathcliff?' I said.	p
para	A nod was the answer.	p

Data mapping

Semantic XML tags are readily mapped to other markup or data structures. A common way such mapping occurs is to allow XML tags to define XHTML classes, so that rich semantic structures are not limited by XHTML's smaller tag set.

The example below shows how this might work. The XHTML tag `<h1>` is used twice, once with the class `"title"` and once with the class `"chapter"`. The visual presentation of each class is defined differently in the CSS.

XML	HTML	Style sheet (CSS)
<title>	<h1 class="title">	h1.title {bold; capitals; 3em}
<author>	<h2 class="author">	h2.author {2em}
<chapter>	<h1 class="chapter">	h1.chapter {center; bold; 2em}
<para_1>	<p class="first">	p.first {no indent; 1em}
<year>		strong.year {bold; 1.2em}
<para>	<p class="next">	p.next {indent first line; 1em}

Thus in an ebook of this content, the title page will use `<h1>` for its first heading, and each chapter will also use `<h1>` for the first heading, but the two classes of `<h1>` will look distinct. Both structural and semantic integrity are retained.

Editing for XML and XHTML

Although the previous examples have shown the tags or coding, editors working in digital markup languages rarely need to do actual hand coding. Rather than learn coding (unless they want to), editors need to:

- focus on the content and structure, not design
- understand how design will be applied (eg through CSS)
- understand XML or XHTML context of their editing environmentp[repare "clean" files with correct markup or ready for conversion to markup formats – "clean" means properly styled and correctly tagged (with opening and closing tags).

Editing environments 1: Microsoft Word

MS Word is still the most commonly used authoring and editing tool in business and publishing.

To work with markup languages, all an editor needs to do is use Word templates and styles – both paragraph and character styles. This is something an editor should already be doing as a matter of best practice.

A properly styled Word file can be imported into layout software (eg InDesign) or into HTML or XML publishing systems.

There are also special Word add-ins for XML, such as MathType for MathML and Inera eXtyles for STM journal and book XML.

Word's default styles and XHTML

Word's default styles correspond to XHTML tags:

Heading 1 = <h1>	Normal = <p>
Heading 2 = <h2>	Quote = <blockquote>
Heading 3 = <h3>	Emphasis = (italics)
Heading 4 = <h4>	Strong = (bold)
Heading 5 = <h5>	Bodybullet = (unordered list)
Heading 6 = <h6>	List paragraph = (ordered list)

Applying Word's default styles to text means it is marked up as XHTML without entering any code at all.

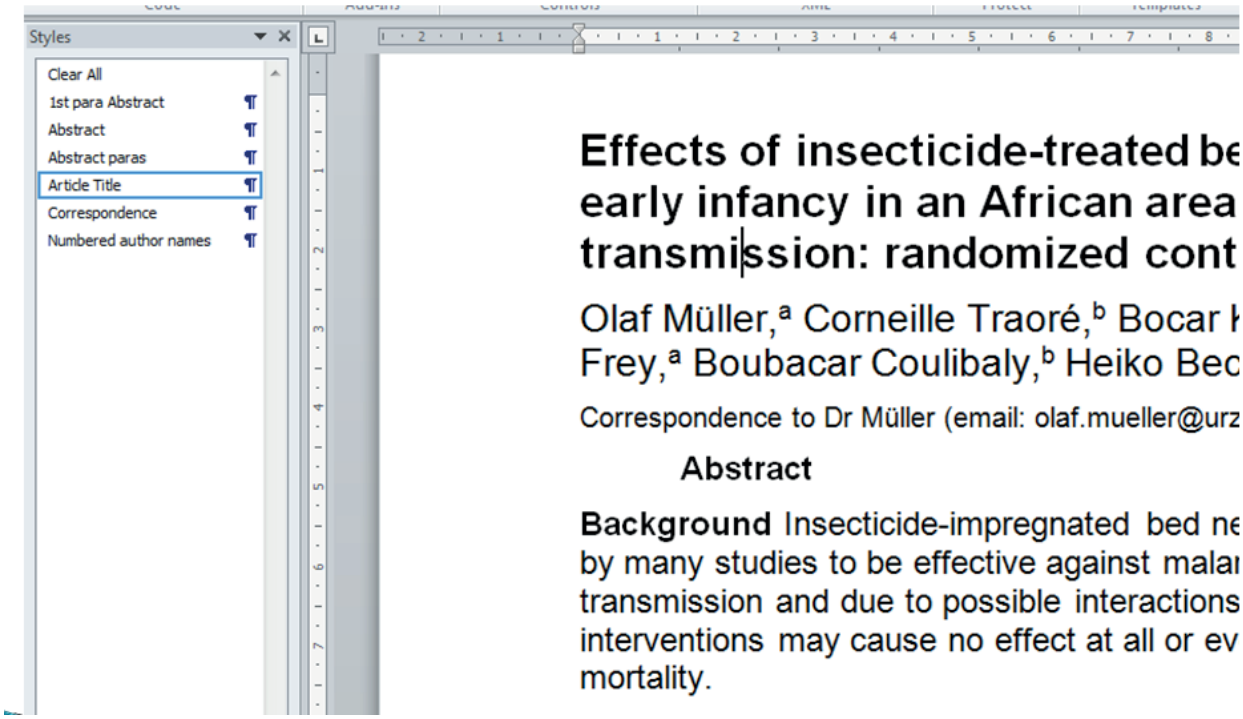
Word styles and XML

Word files are packages of XML files, some of which are XHTML files. The structure of a Word file is defined according to definitions in "Word-XML", and any Word styles become Word-XML elements. Word-XML is rarely used on its own. Instead, Word-XML elements are mapped to elements in other XML schemas, or to style names in other programs (eg InDesign).

When Word is used for import into an XML system, style names are not Word's default styles but new styles with *meaningful* (semantic) names.

Importantly, the names must be appropriate to content structure, and *not* style names that describe the formatting.

In the example shown below, a journal article prepared in Word uses style names that correspond to the element tags in the JATS DTD. The first heading is not given the default style Heading 1 but a style called Article Title. When this context is converted from Word to JATS, the style name will be mapped to the tag <article-title>.



Using Word for markup languages

- Use paragraph and character styles for *all* formatting (good practice anyway).
- Only use Word's default styles if the document will be used for HTML purposes, such as websites.
- Otherwise use semantic, meaningful style names to facilitate correct style mapping to the correct XML elements or XHTML classes. That means knowing your XML context:
  - Styles for journal articles use JATS elements.
  - Styles for user manuals use DITA elements.
- Avoid fonts and glyph issues by using standard fonts eg Arial, Times New Roman.

Editing environments 2: Browser-based CMS

CMS stands for "content management system" and refers to a system in which content (eg text, image, video, audio files) is stored in a database and managed via an interface – usually browser-based. "Managed" can mean any action such as creating, editing, deleting, arranging and publishing.

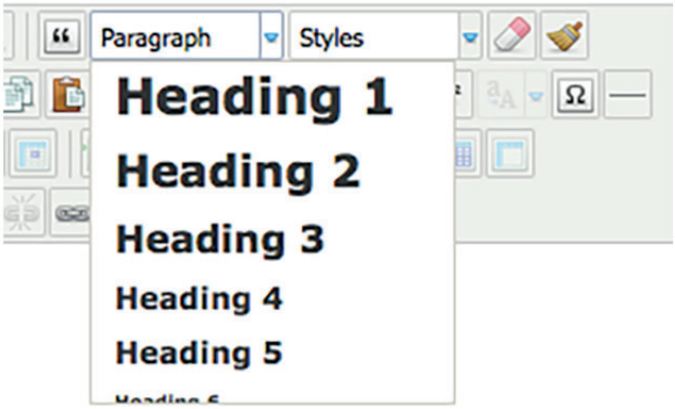
CMSs are used for websites, Intranets and publishing to file formats including PDF, EPUB, XML, XHTML and XML.

Examples of website CMS are Wordpress, Joomla and Drupal. Email examples are Constant Contact and Mailchimp.

CMS text editors

Most CMS text editors provide a tool panel similar to MS Word's tools. They also present a WYSIWYG view by default, and an option to view the code markup. However, they do not usually allow access to CSS code for you to change the presentation within the CMS.

In a good text editor, you don't need to know HTML at all to apply structural markup. For example, the Joomla default text editor provides a drop-down list of all the structural tags for headings and paragraphs (shown below).



Other CMS text editors have a very limited toolset. For example, the Wordpress default text editor has no heading styles. The only way to apply different levels of headings to your text is to enter the code view and manually insert the heading tags.

This is where using Word styles in your source document can really help.

Importing Word into CMS using default styles

A Word document formatted with Word's default paragraph and character styles and copied and pasted into a Joomla or Wordpress page will retain its structural markup. The visual styling comes from the stylesheet (CSS) stored in the system.

For websites using Joomla or Wordpress, all the editing can be done in Word by an editor, and the content then copied into the CMS text editor by a web manager, with no further manipulation needed.

Word to CMS: custom styles

The same principles can be applied using custom (rather than default) styles, as long as the CSS has defined the presentation for classes with the same names.

Editing environments 3: Specialised tools (Sigil)

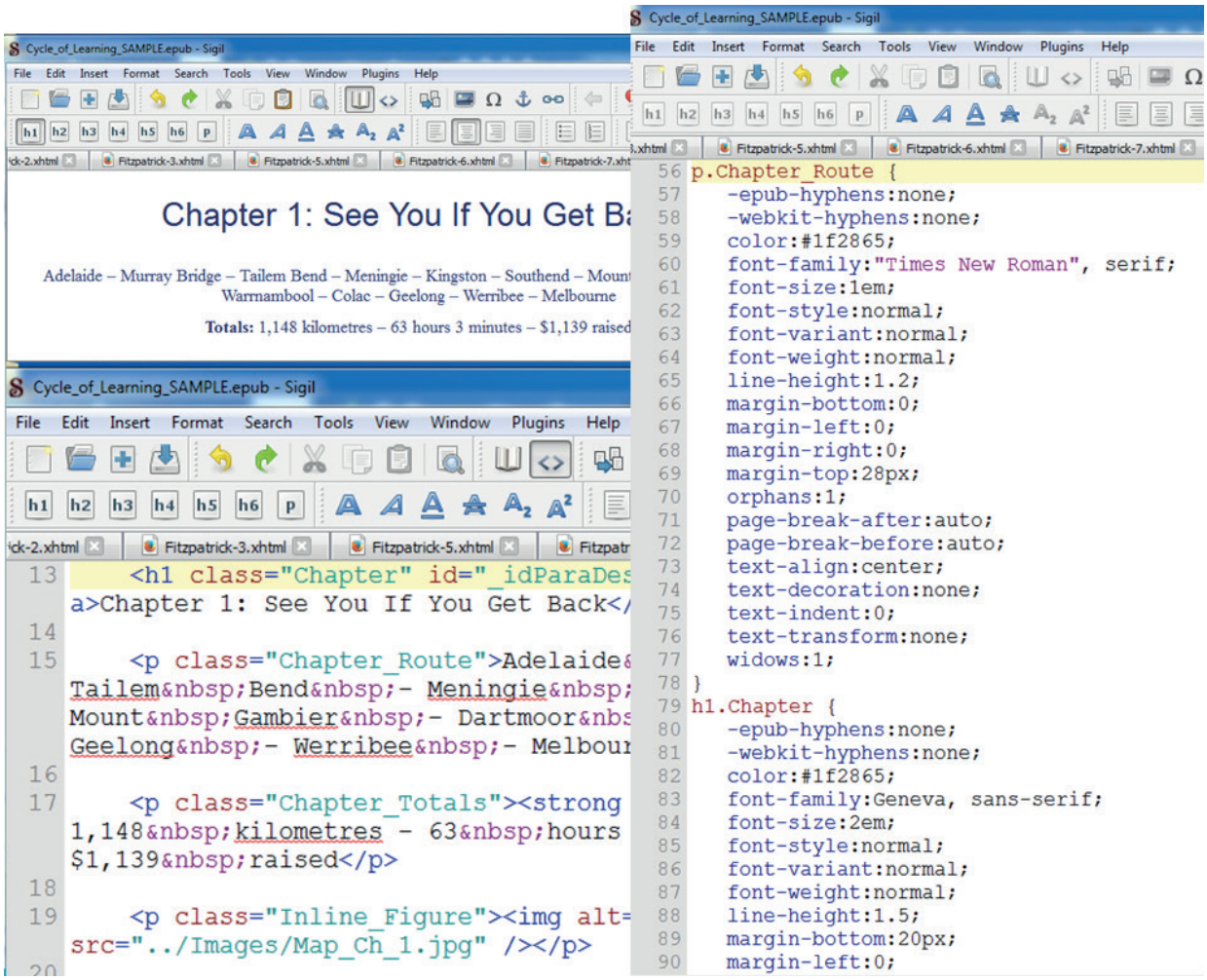
Sigil is a free dedicated authoring, editing and validation tool for ebooks conforming to the EPUB 2 and EPUB 3 standards.

Sigil provides:

- a WYSIWYG authoring and editing environment (much like a web CMS)
- code view, *including* style sheet (CSS) access
- metadata editing tools
- an internal validation tool.

There is not much tech support but it is a friendly learning environment – ideal for getting accustomed to working with XHTML markup and CSS for ebooks.

The screenshots on the next page show a chapter in an ebook in the WYSIWIG view, the code view, and the CSS file that defines the presentation of the text. For example, the heading "Chapter 1: See You If You Get Back" is tagged in the code view as `<h1 class="chapter">`, and in the CSS you can see the coding for the element `h1.chapter {`



Editors and style sheets

Editors are not expected to do the style sheet coding (design) but should:

- understand how HTML markup works with CSS
- be able to investigate the code and troubleshoot incorrect markup
- know enough to communicate more precise instructions to designers and programmers. Compare the following instructions:
  - "make chapter titles bigger" vs "make <h1> font size three times bigger than <p> font size".
  - "underline all proper names and make them blue" vs "create a new span.name class with color: blue and text-decoration: underline".

ML-based publishing systems

Markup language-based publishing systems fall into two main categories:

1. **All-inclusive software**, eg InfoGrid Pacific, that:
   - handles all or most XML / XHTML data tasks (editing, validation, export to production formats)
   - is supported by a single software developer (external vendor or in-house IT department).
2. **A series of software programs**, eg Typefi, each handling a distinct subset of the tasks, that are linked to a greater or lesser degree via XML and are supported by several developers (external or in-house) who may be working together or in isolation.

Whichever system is in use, the question an editor needs to ask is: at what point do Word files enter the system?

Advantages of ML systems (why publishers love them)

- Streamlined publishing processes.
- Markup takes care of formatting.
- One content source for multiple formats.
- Less duplication of content, with less unnecessary variation.
- Content is exchanged or transmitted, not copied and pasted or re-entered.
- Improved accuracy and efficiency.

Problems with ML systems (why some people hate them)

- Poor understanding of editing and publishing requirements among IT staff and vendors.
- Poor understanding of technology and XML capabilities among system users.
- "Out of the box" systems too inflexible.
- Custom systems too complex and expensive.
- Unrealistic and overambitious projects – expectation that XML will solve everything.

Evaluating new publishing systems

Editors involved in implementing a markup language-based publishing system need to look closely at several items.

1. Content, especially structure and industry requirements.
2. Processes for authoring, editing, proofing, correction, production.
3. Primary purpose for the system, eg to facilitate XML file exchange, deposit, and archiving, or to gain production efficiencies.
4. Impact (positive and negative) on staff, budget, IT systems etc.
5. Support: IT help, editorial training, troubleshooting; installation only or ongoing; in-house or email only.

Whenever new systems that impact editorial workflows are being implemented, it is vital that editors are involved. That means editors must know enough to be part of the conversation.

Editing skills: "knowing enough"

Editors need particular skills to work effectively in modern editorial and production environments. An editor needs to:

- apply on-screen markup – the method will depend on the editing tools in use, but applying Word styles is essential
- edit content independent of presentation
- interpret content according to structure
- understand markup and validation processes
- understand production outputs (formats) and processes, eg style sheets
- proof formatted content and identify markup (code) issues by looking at the source code or communicating the problem to IT or design staff
- know XML schemas relevant to their domain.

Myths about editors and markup

"Tagging is not an editor's job"

Markup is a core editorial task, whether on paper or screen. Most markup is done in WYSIWYG systems and is as simple as applying styles.

"Editors have to become typesetters"

In most markup language-based systems, typesetting is usually automated to a greater or lesser degree.

Designers create templates and style sheets, and programmers write the code that applies these to marked-up text.

The editor works on the content and applies markup to the content files (often as styles) for import into the typesetting applications. This is not typesetting, any more than marking up hard copy was.

Editors do need to understand the typesetting contexts in which they work – just like they always have.

Practical workshop for editors

Editors wishing to gain hands-on experience with using markup in applications like Word and Sigil can enrol in the workshop *Editing within and for markup languages* (formerly known as *Decoding XML: a practical guide for editors*).

The next workshop will be run for the Society of Editors (NSW) in Sydney in November 2015.

To arrange other times or a tailored version of the workshop for in-house editorial training, contact Linda Nix at Golden Orb Creative, via email linda@goldenorbcreative.com or mobile 0417 041 744.