
AODC Conference 2010

Dave Gardiner, Red Lettuce Communications, <info@redlettuce.com.au>

Table of Contents

Overview	1
Matthew Ellison (Matthew Ellison Consulting)	1
Dave Gash (HyperTrain dot Com)	2
Tony Self (Hyperwrite)	3
Joe Welinske (WritersUA)	4
Monica Allen (Talent2)	4
Suchi Govindarajan (MYOB)	5
Sarah Maddox (Atlassian)	5
Dr Alan Burton (eGloo)	5
Frank Munday (Australian Federal Police)	6
A walk through Google apps	6

Overview

The 13th Australasian Online Documentation and Content Conference was held in Darwin, 12–14 May 2010. Presented by HyperWrite and WritersUA, some 30 technical communicators attended to hear, and share experiences in, the latest developments in technical writing. All talks were in the context of writing and presenting content for computer software online help documents. XML technologies featured prominently, the Darwin Information Typing Architecture (DITA) XML schema being particularly associated with technical communication. It was definitely an eye-opener into how editing and publishing could develop over the next few years.

Matthew Ellison (Matthew Ellison Consulting)

'Turning search into find', Matthew's first talk, covered some innovative approaches to searching for content on the web and how to structure content to make it easier to find. Search is not necessarily the most effective tool to find information, but it is the tool that users prefer to use. Google Suggest has predictive searching, so that as you type each letter of a word to search, it will predict what words you might be looking for; so with a few keystrokes you are able to select from a list of suggested words. The suggested words that appear are the most popular words related to problems that users want answered; i.e. help issues. Google will suggest not only words and phrases beginning with the letters typed in, but also partial words where typed letters appear in the middle of words or phrases. Boolean words can be used in search terms, such as to search for specific text that is NEAR other text. There is a need to balance the search functionality with simplicity/ease of use; strong features of online searching (the websites of a bookstore and publisher were demonstrated) include predictive searching, simple user interface, ability to enter search terms for title or author, and display of matches within titles. Faceted searches classify information by specific characteristics (e.g. parts of words or phrases), and allow users to explore information by narrowing down a search and refine results from the general to the specific; e.g. a search for wines can result in broad categories – e.g. varieties or regions or price – from which users can select subcategories to refine a search until they find a specific product. Some factors to help turn search into find are: stop words – to exclude specific words from search matching; a facility to exclude specific topics from a search; search result synopses (show a key extract for each search result); boolean search; phrase matching (entering phrases as quoted keywords); fuzzy matching (offer close matches or misspelled words as results); search filtering and faceted search (categories and subcategories on a screen); ranking of search results; metadata (where

topics are found even if the search word is not in the visible text); and predictive search (e.g. Google Custom Search can be added to any website; PredictAd provides suggested search queries).

'What kind of assistance do users really need?' outlined various ways that help topics can be structured and presented to assist users with software or online procedures. Help topics can include feature-oriented, task-oriented and frequently asked questions. These vary in the amount of information presented (whether as brief points or extended discussions), the use of numbered steps to solve a particular software problem, and how topics can be found using questions (the most common being 'How do I...', 'Can I...' and 'Why can't I...' questions). Among the traps for writers of user assistance are: simply transcribing information provided by developers; explaining the obvious; and maintaining consistency for the sake of consistency. The Skype Help portal is an example of how a text search field is balanced with a listing of 'How do I' questions, and related search terms are featured as a lower priority in a different part of the computer screen; search results have a clickable topic heading to display step-by-step procedures, as well as a high-level summary. Other techniques to present search results include the PayPal Virtual Agent, which presents a conversation-style interface.

'Help authoring tool comparison' presented the functionality and interface usability of several popular help authoring tools (HATs). Using HATs automates the generation/compilation of content for browser-based output, is useful for single-sourcing and provides indexing. These can be in the form of HTML-based authoring tools, XML-based authoring tools, wikis and blogging tools. Advantages of using HATs include that they are specifically targeted towards user assistance, there are 'off the shelf' solutions, they provide a one-stop solution for multiple outputs, and they require little or no knowledge of the underlying technology. Disadvantages are that they can lock you into proprietary technology, and have limited capability for content management and re-use. The HATs presented for comparison were Adobe RoboHelp, Adobe Technical Communication Suite (TCS2), Author-it, ComponentOne Doc-to-Help, EC Software Help & Manual, MadCap Flare and WebWorks ePublisher. Functional features that most of the HATs have in common include: generation of table of contents and index, MS Help output, browser-based Help output, conditional content, snippets/embedded topics and DHTML drop-down menus (concertina-type menus). The HATs were tested for how they handle workflow, user interface and usability, key strengths and weaknesses, and how they generate their own Help topics. Efficacy of workflow examined how each HAT handled input formats (e.g. Word, FrameMaker, RTF, PDF, HTML, DITA, RoboHelp), storage formats (XHTML, XML, database) and output formats (Word Help, Word/PDF, Adobe AIR, DITA, various Help formats, ebook and wiki formats). Ease of use, complexity for learning and the conversion between formats featured in the comparison, and whether documents were usable in a software-specific proprietary format or a standard format that could be shared between different software packages.

Dave Gash (HyperTrain dot Com)

'The power of controlled language' introduced the concepts and application of controlled language (e.g. plain English) to ensure that any text can be understood by all users, regardless of the degree of familiarity with the English language. Controlled language uses a limited set of standardised words to make technical text easy to understand. It is based on natural language that uses simplified grammar and style rules to reduce ambiguity in meaning. Having a limited set of approved words and meanings, it aims to reduce complexity and confusion of technical documentation. Controlled language aims to improve language translation, makes single-sourcing more efficient and authoring faster; documents are easier to understand, as comprehension is improved. However, there is resistance from writers and editors to adopt controlled language due to the perceived loss of nuance in text; organisational management may see software and training expenditure as hurdles. Simplified Technical English (STE) as a controlled language is encapsulated in the international standard ASD-STE100 specification (derived from aerospace and defence), is actively developed, and can be used (is extensible) in new hardware and software environments. Some of the grammatical rules of STE are based on the principles of good writing, such as: use the active voice; use articles; use simple verb tenses; and write short sentences and paragraphs. Adopting STE includes the broad issues of acquiring the STE standards, developing a 'corporate' dictionary to add subject-specific words, training

writers and editors, and using software tools (e.g. Textanz Concordance, Acrolinx IQ, Tedopres HyperSTE, MAXit Checker, SMART Text Miner).

'Introduction to DITA conditional publishing' outlined how instructions can be included in XML files to process text content and modify the output of documents (e.g. specify which parts of the content to publish). The talk discussed how XML tag attributes [presented in this talk as 'metadata'] can be used to assist searches for specific content, prepare keywords for indexes and control what is published. These attributes allow a fine degree of conditional processing (e.g. include or exclude specific content from a document) which, with DITA, involves setting up an element layer, topic layer and map layer for each document (i.e. apply attributes to specific content, which affects, for example, what particular content should be displayed in a document). Then, processing conditions are written in a 'ditaval' file that states what attributes should be selected (included or excluded) for a 'build' (create an output document) based on the attributes defined in the previous step of setting up layers. Various DITA-aware XML authoring and publishing tools are available to make it easier to create coding for conditional processing.

'Creating auto-magic TOCs with XSLT' discussed how to generate tables of contents (TOCs) by coding XSLT templates. Text that is marked up in XML documents refers to XSLT stylesheets; in much the same way that with web design, HTML documents refer to CSS files to control the style of output text. XSLT is a 'declarative' programming language that applies actions to textual content in that when XSLT documents are processed, they treat the content as 'when you find something, do this'. In handling complex documents (with multiple headings, heading levels, paragraphs and lists), the use of multiple XSLT templates apply different styles to different parts of a document, and nested XSLT templates allow differing degrees of style processing based on a hierarchical structure of templates that refer to other templates depending on what type of styles to apply to different text. When the concepts of both multiple and nested templates are used together, you could specify broad text styles in a 'parent-level' template (e.g. such as all body text in the whole document to be a particular typeface and font size) and more specific styles in a 'child-level' template (e.g. level 2 and 4 headings to be bold italic). XSLT 'modes' templates is a third type of processing code that specifies the order in which to process information contained in XSLT templates. Normally, templates are processed from the top-down, but modes enables processing to jump from one part of a template to another. This allows the textual content to be generated in an output format, and then a second pass to generate a TOC for the headings in that text.

Tony Self (Hyperwrite)

'An update on DITA features, tools, and best practices' focused on DITA authoring and publishing tools and their functionality, and development of the DITA 1.2 schema. Authoring tools include Arbortext Editor, Framemaker, XMetaL, XMLMind XML Editor, Serna, DITA Storm and <Oxygen/>. These tools can operate within an integrated content management system (CMS) as plug-in editors, with CMS packages like Bluestream XDocs, DITAworks, Ixiasoft DITA CMS and Sibersafe among the more popular. Reviewing tools that allow multiple authors to modify content include XMetaL Reviewer. Publishing and rendering tools that generate document output include MadCap Flare, Adobe RoboHelp, Antenna House Formatter, XMLMind DITA Converter, WebWorks ePublisher, WinANT Echidna and Altova Stylevision. These tools have varying levels of support for editing and publishing functions, and formats for import and export (e.g. Help files, epub). The DITA Open Toolkit is used by many publishing tools to render documents for output, which includes the generation of TOCs and an index. DITA 1.2 is being developed with a less rigid structure of elements, new elements, and more flexibility with constructing glossaries from acronyms and terms in the text. Among best practices being touted is the need to choose a tool that is compliant with the DITA schema, choose DITA for the right reasons (e.g. improve re-use of content), the need for more than one tool as part of a 'toolbox' for authoring and publishing, and the possible need for specialised assistance with XSL and graphic design.

'The wonders of SVG' introduced the scalable vector graphics (SVG) format for displaying graphics in web browsers and on mobile devices. SVG is a text-based format to define images with vector shapes, text

and raster graphics. High-quality graphics with small file sizes can be produced, and as an XML language can be used with many applications on different computing platforms. Simple animations with audio can also be generated using SVG. Some advantages are that SVG can be embedded in PDF documents and text in graphics is searchable. But not all browsers fully support the format and some SVG editors can produce invalid coding. Raster graphics and video can be modified using JavaScript, and SVG can contain Flash and MP3 data. Browsers currently offer full support (Opera), good support (Firefox, Chrome, Safari) or poor support (Internet Explorer) for SVG. Editing tools to create SVG include Visio and Batik.

'**WinANT Echidna – The DITA Open Toolkit made easy**' demonstrated an open source interface that makes it easier to use the DITA Open Toolkit (OT) for authoring and publishing DITA XML files. Configuring and using DITA OT necessitates running processing instructions at a command prompt, and hand coding files. WinANT provides a windows graphical interface to install DITA OT and run transformations, while supporting relevant plug-ins. Capabilities include ditamap transformations, conditional publishing, handling graphical layouts (e.g. report covers) and dynamic help topics. The Integrator function helps to install plug-ins, XSL styles can be modified for PDF2 documents, and can run alongside a DITA editor.

Joe Welinske (WritersUA)

'**UA design and implementation for mobile apps**' presented examples of the graphical design of user assistance (UA) applications and the implications for improving usability. UA has relevance to apps, in presenting complex topics that can be viewed and understood easily (such as interactive menus). Design issues include using simple terms that people understand (e.g. tap, type, select, pinch) and defining viewports for resizing web pages to mobile device screens. Layout of screen content ('real estate') can be controlled with HTML and CSS code. Resources and galleries to support the design of help content for iPhone apps include various iPhone developer forums.

'**Optimising the googleability of your content**' examined how content and metadata can be structured to improve the chances of being found in searches. Getting content on the public web is the first step (with web sites, web-based help and PDFs among the better formats for searches), although some Help formats and Flash are difficult to google. Security issues that affect whether content can be found include firewalls, content that requires a login, and determining what content really needs to be secure anyway. Google Search indexes web content with smart algorithms, ranks content and matches search queries against the index to return results according to priority. High rankings can be achieved by ensuring links to your content, following webmaster guidelines and considering Google Ads. Search engine optimisation can be improved by using sitemaps, metadata (title, description, keywords) and registering your site with Google. 'Community building' can improve searchability, through creating Facebook pages, YouTube videos and LinkedIn group content. Finally, Google Custom Search can be implemented on websites by registering with Google and adding code for a search box.

Monica Allen (Talent2)

'**Managing a documentation project**' was a case study in developing content and managing the production of SAP training documentation. The project for a multinational natural resources company was run out of Singapore, with development work split between staff in Melbourne (work instructions) and India (guides, workbook). Initial planning work set the start date, goals and scope; identified stakeholders; and organised a communications plan, schedule and staff roles. Clearly defined roles assisted in finding elearning designers and specialist writers. The documentation design stage entailed identifying topics, creating topic hierarchy and determining the delivery methods for elearning and facilitated sessions. The production stages progressed as follows: research (with functional specifications and storyboards), develop, review accuracy and usability (in consultation with subject matter experts), approve, track progress and report on outcomes. Among some of the problems were delays with staff obtaining computer user accounts, lack of engagement of small to medium enterprises (SMEs) and a high turnover of project managers.

Suchi Govindarajan (MYOB)

'Who's afraid of the DITA wolf?' outlined a DITA user's experiences with devising and delivering an entry-level training course in DITA for organisational staff. The barriers to learning a new form of structured authoring can best be overcome by avoiding the theory and jargon (which can be convoluted and confusing), and just start authoring. (DITA users do not need to know XML to begin topic-based structured writing.) You do need to know the basic theory of elements; attributes; and the barest idea of concept, task and reference. You do not need to know the concepts of specialisation, customisation, inheritance or DTDs (document type definitions). The 'wolves' along the road to learning are DITA Open Toolkit (which has poor documentation) and letting small problems lead to not wanting to learn. The 'DITA in a day' workshop that was devised resulted in writers being able to write a concept and task, and create a DITA map to pull them together, within an hour. Writers discovered elements on their own, and guessed what they were used for. The type of XML editing tool was critical to fast learning, with a DITA-aware editor being most useful. A quick introduction to the workshop covered elements, attributes, topic-based authoring and basic structures. Start the session by showing a very good example of a well marked-up document, then learn simple things such as task/concept/reference, notes, cross-references, hyperlinks and tables. Refer to the DITA specification to strengthen understanding of concepts. Publishing the XML coding to documents was done with WinANT and XMLMind DITA Converter.

Sarah Maddox (Atlassian)

'Engaging your readers in the documentation' showed the advantages of writing and presenting content to retain readers' interest, thus better ensuring more customers buy your product. People need to be engaged to buy more and be satisfied with the documentation product. Ways of engaging readers include comments on documentation web pages, links to readers' blogs and articles, open editing of wiki pages, and presenting tips using Twitter. Linking to readers' blogs can be useful because they can be of a high quality, but research the blog before adding a link. Open editing of wiki pages is acceptable, as long as permissions are set for users adding to documentation and that technical writers monitor the quality of contributed content; this is considered a useful addition to create documentation, and fills knowledge gaps yet requires little maintenance. Twitter was used to distribute release notes for documentation, with three tweets a day for regular news and tips for users, and 10 tweets a day sent for major releases. A 'doc sprint' session was held to get initial documentation written, where a small team of technical writers was shut up in a room until documents were written; with occasional fun activities, this is a highly productive forum to produce quality documentation.

Dr Alan Burton (eGloo)

'Converting to structured content' discussed the challenges and problems of converting electronic documents to XML format, as part of changing a publishing system to XML workflow. Legacy content (i.e. existing documents in various formats such as Word, Famemaker, desktop publishing applications, paper) is unstructured, and converting documents to a structured format requires considerable planning to establish specifications and standards. Decide on who will do the conversion – in-house, outsource, use an overseas (e.g. India, The Philippines) conversion service? You need to analyse the content of the documents to convert by ensuring you specify the exact styles for every conceivable type of text, foreign and mathematical characters, equations, tables, graphs, figures, etc. in all your documents – who will do this analysis? Analyse the documents before you select an XML DTD, as the type of content will affect the DTD you choose. Prepare a sample document – which is provided for the conversion people to follow – that contains all examples and variations of content. Consider how metadata, indexing and cross-referencing should be included. This then enables you to select a document type definition (e.g. DITA) to use as your base XML format for all documents, and the XML software that supports the chosen format. Write Requirements Specification and Markup Specification documents, which specify the exact XML markup to use with each

type of content; this should include detailed examples. An additional Mapping Document must state what text styles (e.g. headings, paragraphs, lists) in your documents should correspond to what XML tags are used to mark up the content. Problem areas include table markup, graphics and graphic formats (multiple graphics formats may be required), cross-references, non-standard characters, metadata and forms. Converting from unstructured to structured content can be done manually and/or as a semi-automated process with Adobe FrameMaker, Word to FrameMaker (as long as styles are set up in Word), Word to XML (although none of this conversion software gives a good result; lots of customisation is required), or desktop publishing to XML (InDesign is style-oriented and may not be amenable to structured authoring).

Frank Munday (Australian Federal Police)

'I can't spell ambliance!' was a light-hearted look at the scourge of misspelled and misused words, redundant words, mixed metaphors, and concocted words that can pervade technical writing and blogs. Among the advice was to use metaphors sparingly, and be vigilant to remove extraneous words and phrases that could cause confusion and obscure meaning.

A walk through Google apps

An Experts Panel discussed the potential of several Google applications for creating websites and sharing content. Google apps are office software products that are delivered online at no cost ('Standard' access) or for a nominal subscription fee ('Premier' access), depending on the functionality, storage space and access required. The apps are Gmail, Google Calendar, Google Docs, Google Groups, Google Sites and Google Video. Google Sites can be used to set up public websites, share information on a secure intranet and collaborate on a team project; password-protected sites and the ability to store various forms of content (e.g. wiki, blogs) are features. Google Docs can be useful for sharing documents (e.g. Word, OpenDocument, RTF, spreadsheets, HTML) and collaborative authoring (and see changes in real time). Gmail can incorporate your own domain name and has support for the Blackberry device. Google Groups can handle mailing lists and discussion groups.

Resources from the AODC 2010 Conference can be found at <http://www.aodc.com.au/resources.aspx>.

This article was prepared using the DocBook 5 stylesheet distribution [<http://sourceforge.net/projects/docbook/files/#files>].