Template design issues for word processors and (possible future) EPUB export

[This document is a re-post of one for the jiscPUB project – please comment over there: http://jiscpub.blogs.edina.ac.uk/2011/08/03/template-design-issues-for-wordprocessors-and-possible-future-epub-export]

This document is a collection of notes on how to design word processing templates for creating EPUBs – particularly theses. It's probably not very interesting as a general read. The intended audience is support and technical staff who are working with theses, and preparing for ebook creation projects. It may be of use to projects following from jiscPUB, particularly in the area of thesis management submission and deposit where theses are required to be published in HTML and/or EPUB. These notes are incomplete – this is not a full "word processing for theses" book and it does not provide an answer of how to actually create EPUB theses of high quality from word processing documents, although I produced some promising demonstrations of the potential during my work on this project.

To make theses produced with a word processor available as EPUB it is a given that the word processor, or some other application which can read word processing documents needs to be able to produce good quality HTML. Given HTML EPUB can be created even if the word processing package or content management system being used is not capable of exporting EPUB natively. As Liza Daly notes in the final report for this project that's difficult to achieve from arbitrary word processing documents, which is why it is useful to design a template, documentation and training that helps users to choose features in their word processors, such as using defined styles rather than direct formatting.

I was involved in a word-processor based web publishing project at the University of Southern Queensland from 2004 to 2010. The project, the Integrated Content Environment produced some templates, toolbars for creating documents, and HTML conversion code all released under an open source license. I refer to that project a lot here, as it dealt with many of the relevant issues in setting up word processors for academic use, including fairly comprehensive documentation about how to do things the right way. There is a fork of the project on Google Code which I added to during the jiscPUB project.

This document takes a general look at template design and provides some specific examples and advice for two applications, Microsoft Word and OpenOffice.org Writer (including the new LibreOffice fork and the other derivatives); many of the issues are the same for other word processing packages but this project didn't have the resources to explore all the available options such as Apple's Pages and Google docs. Another option is the open source LyX word processor (or document processor as the creators call it) which, with some training and template development may suit some candidates. But note that it would need to be run in a **very** well-supported environment.

In this document I refer to the thesis template provided by the university of Edinburgh and use it as the basis for some examples.

Templates vs "document prototype factories"

Templates are a starting point for creating different genres of document. When properly installed, they allow users to choose something like `File / New / From template...` and to pick the kind of document they want; a thesis chapter, a paper, report or blog post. But they suffer from several usability and maintenance issues in today's computing environment.

- If you click to open a template, it spawns a new document. In my experience users tend to save these new documents wherever they work normally – maybe in a shared drive, often on the desktop, and leave the template where they downloaded it. So the most likely place a template will end up living is in the Downloads or Desktop folder – where it is not subject to version control or management.
- In OpenOffice.org the template system is arcane and difficult to navigate – it is possible to import a

template via the user interface but it is complicated.

My advice here is not to attempt to distribute templates unless it is possible to do so via something like a standard institutional desktop, but to make blank prototype documents available for download from a content management system or a shared directory, and to put in place managed processes, automated if possible for creating the prototype documents; creating something along the lines of a 'document prototype factory'.

If you decide to maintain a family of document templates:

- Try to share as much as possible between document prototypes/template, including style names, and if possible the same fonts and margins to reduce maintenance overhead.
- Maintain the core styles and common elements from the templates in one place – a 'master' template.
  - When making changes make them in the master template and then import the changes into the other templates/prototypes.
  - Consider automating production of sets of styles using macros or by producing the raw XML for .docx or .odt files. The ICE system, for example, contains macros that create a complete set of styles on demand using default settings. This means (if the macros work and I'm not sure that they do 100%) that a new template can be created by setting margins, and the font and spacing for a couple of base elements, and having the machine generate all the rest.

Granularity

One of the fundamental choices to make in designing templates for long documents like theses is whether to manage the document in one long file or to break it up into multiple chapters.

Historically, it was important to work on compound documents for performance reasons. These days, performance is probably not a major problem, with most computers having plenty of RAM, but there are still reasons why compound documents make sense, for example where a resource is to be assembled out of a range of source documents, or other objects. It makes particular sense in collaborative environments, where multiple parties are working on a project and editing different chapters. Theses are not usually meant to be collaborative (although that might be changing) but in the absence of collaboration infrastructure which can manage comments from a supervisor, sending off chapter one to a supervisor to add comments while the candidate works on chapter two allows for simpler management than mailing off the whole thesis for comment, and then having to integrate the two versions.

The major problem with the compound approach is when it comes time to join the thesis into a single final product for printing.

Microsoft Word has long had a reputation for poor performance in managing master documents. I have not checked this in detail in the latest version but I would urge any template project to check its performance carefully before relying on it. The simple approach of copy-pasting multiple things into one once the thesis is finished, as recommended in the help text in the Edinburgh thesis template is possibly the most reliable but it can be time-consuming and small differences in formatting that have crept in to the various chapter documents can cause problems.

The ICE project used compound documents because its focus was course documents which were authored by multiple parties, but our initial experiments with OpenOffice.org master documents assembled by computer program were not a success, so we settled on an approach which automated copying and pasting things together, according to a table-of-contents-like manifest, to produce a final compound file, avoiding all sorts of complexities to do with differences between page layout and changes to styles, which can occur by accident.

With the rapid rise of ebook readers and a shift away from paper-based publishing, we should, in the academy be considering that thesis submission is a web-based process, possibly with EPUB as a container format, with thesis projects taking a few years to complete, the time for projects that consider how thesis authoring and submission should work is **now**.

How to set up a master document

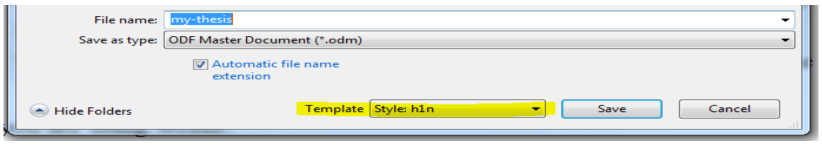
In this section I have some sketchy instructions for setting up compound theses via master documents, note that these instructions are a starting point only.

In Writer you can turn a long document into a master document with multiple parts – I put examples of these in the demonstration system for the jiscPUB project.

- First, use styles for your headings.
- Work out which heading style is being used for Chapter headings. In the Edinburgh template it's Heading 1 – in a typical ICE thesis it would be `h1n` or `Title Chapter`. I have used `h1` in the examples.

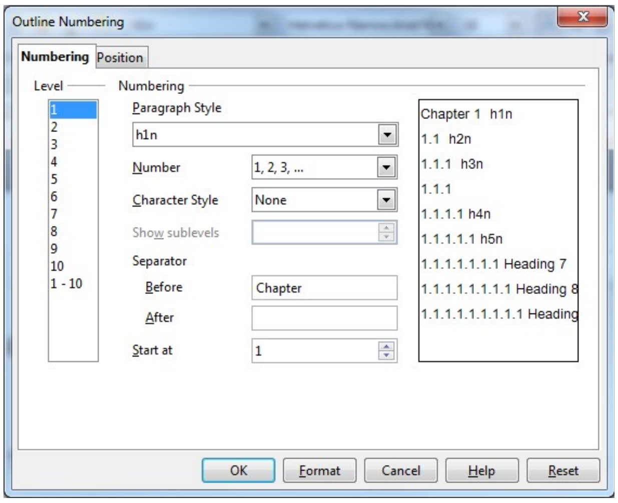
If you are using Writer:

- From the File menu choose `Send`, then `Create Master Document`.
- In the `Template` dropdown choose the style that's used for chapter headings and provide a file name.



File name: my-thesis
Save as type: ODF Master Document (*.odm)
☑ Automatic file name extension
Template: Style: h1n

- Click `Save`.
- The application will create a series of files, one for each block that starts with the chapter style. (Or at least it should – there seem to be bugs in LibreOffice 3.3.3 and the splitting feature didn't work for me). The resulting master document will contain all the front matter text with the chapters included. I recommend moving this to a sub document too:
  - Select all the front matter and Cut it.
  - In the navigator in the Master document, right-click on the first chapter (eg my-thesis1.odt) and choose Insert New Document.
  - Paste the front-matter into the new document.
  - Save the new document as my-thesis0.odt (for example).
- To make the chapters usable as stand-alone documents:
  - Open each document.
  - In Tools / Outline Numbering, set the Start at number for the first outline level to the chapter number.



Outline Numbering

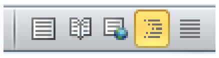
Level: 1
Numbering
Paragraph Style: h1n
Number: 1, 2, 3, ...
Character Style: None
Show sublevels:
Separator
Before: Chapter
After:
Start at: 1

Chapter 1 h1n
1.1 h2n
1.1.1 h3n
1.1.1
1.1.1.1 h4n
1.1.1.1.1 h5n
1.1.1.1.1.1.1 Heading 7
1.1.1.1.1.1.1.1 Heading 8
1.1.1.1.1.1.1.1.1 Heading

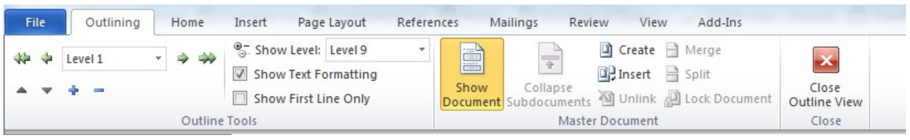
To perform the same trick in Microsoft Word you have to split the document manually.

- Copy and paste all the chapters and the front-matter into a series of files.
- Create a new, blank document with the correct margins and styles.

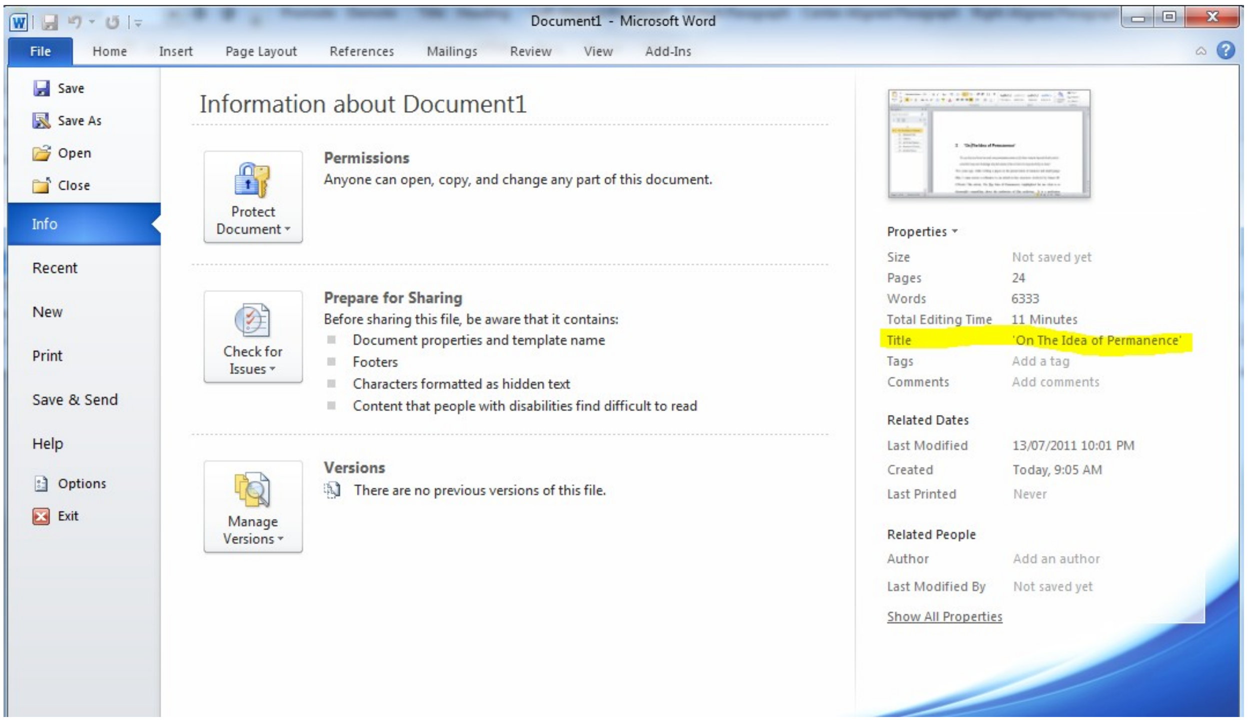
- Switch to outline view via the status bar, bottom right of the document window.



- In the Outlining tab, click Show document (this makes more options appear).



- For each chapter, click Insert, then pick the chapter and click Open.
- To make sure that each chapter has the correct number and title when you are managing in individually:
  - For each chapter, open the file itself.
  - Copy the text from the chapter heading and enter it as the document title in the File tab under Info, Properties, Title.



Information about Document1

Permissions
Anyone can open, copy, and change any part of this document.

Prepare for Sharing
Before sharing this file, be aware that it contains:
- Document properties and template name
- Footers
- Characters formatted as hidden text
- Content that people with disabilities find difficult to read

Versions
There are no previous versions of this file.

Properties

Size	Not saved yet
Pages	24
Words	6333
Total Editing Time	11 Minutes
Title	'On The Idea of Permanence'
Tags	Add a tag
Comments	Add comments

Related Dates

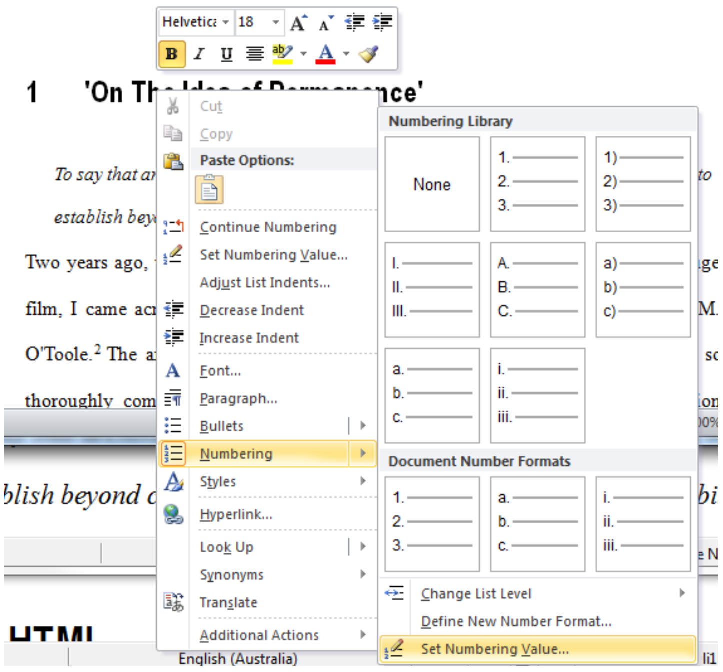
Last Modified	13/07/2011 10:01 PM
Created	Today, 9:05 AM
Last Printed	Never

Related People

Author	Add an author
Last Modified By	Not saved yet

Show All Properties

  - In the chapter heading, right click and choose `Numbering`, then `Set Numbering Value...`



- Choose the chapter number in Set value to: and click OK.

In either Word or Writer you now have a series of stand alone documents that you can edit one at a time, and a master document that contains the whole book.

Converting the files to HTML

Converting the thesis to HTML can now be done either chapter by chapter, for example as a series of posts or pages in WordPress, or by via the master document, with the usual caveats that word processors tend to make poor HTML. One drawback of the approach I have outlined here is that each of the sub documents uses the Heading 1 style as its title so when converted to HTML as a stand alone document has a slightly odd structure. Dealing with this kind of document structure is something for a (forthcoming) wish-list of features for a good quality HTML converter – it should be able to normalise headings in the documents it outputs, and 'do the right thing' with each document delineated by article tags, containing sections. HTML5 has specific rules about document outlines which allow for re-combining content from multiple fragments.

Styles

Styles are one of the key innovations that make word processing useful for technical and academic content. A style is a named bundle of formatting attributes that can be attached to a paragraphs, span of text inside a paragraph and to lists and table structures.

The most basic use of styles (and the only area where there is anything like a cross-application standard approach) is using heading styles to structure a document. Most word processors use `Heading 1`, `Heading 2`, and so on attached to paragraphs as the standard way to create a document outline. That is where the quasi-standardisation ends, though, there are no widely used standards for the other things we need in academic documents.

A thesis template should have, at a minimum styles for:

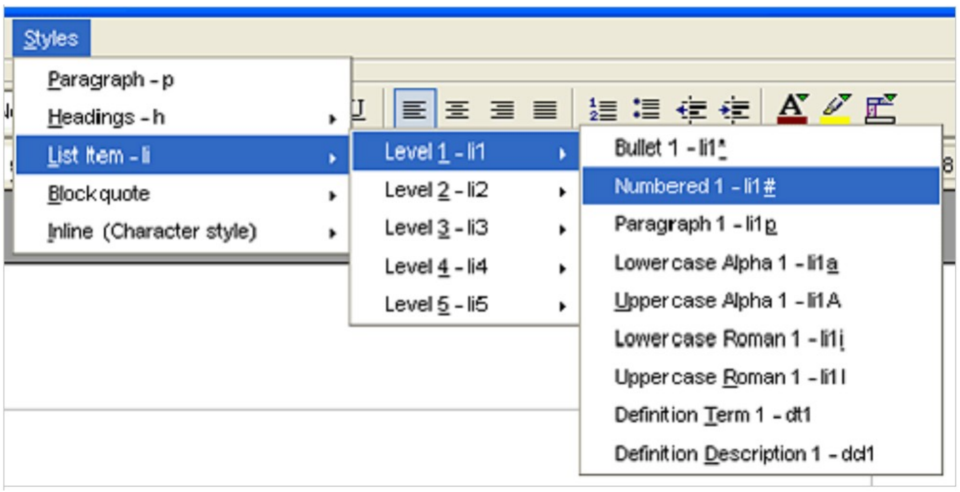
- Headings
- Metadata
- Block-quotes

- Examples
- Pre-formatted code.

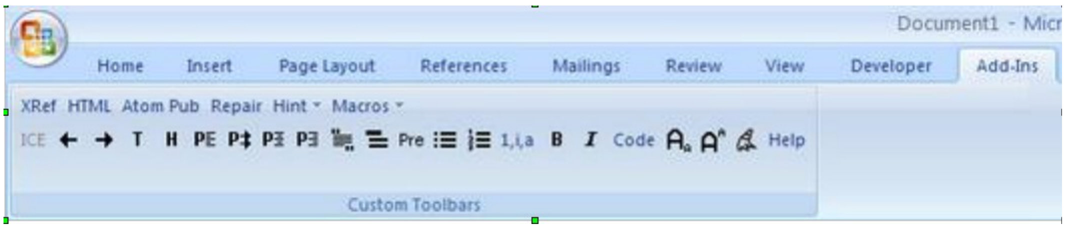
For example the ICE system specifies a set of styles which has been used for several years for producing academic documents at the University of Southern Queensland. The styles are summarised here. In a table, updated from one which originally appeared in an article at xml.com. These style names were chosen to be mostly very short, so they would be easy to see in the interface in both Word and OpenOffice.org, particularly in Word's view that shows style names on the left.

Family	Type	Style names				
		1	2	3	4	5
Paragraph (p)	p-centre, p-right, p-indent*	p				
Heading (h)		h1	h2	h3	h4	h5
Heading (h)	Numbered (n)	h1n	h2n	h3n	h4n	h5n
List item (li)	Numbered (n)	li1n	li2n	li3n	li4n	li5n
List item (li)	Bullet (b)	li1b	li2b	li3b	li4b	li5b
List item (li)	Uppercase Alpha (A)	li1A	li2A	li3A	li4A	li5A
List item (li)	Lowercase Alpha (a)	li1a	li2a	li3a	li4a	li5a
List item (li)	Lowercase Roman (i)	li1i	li2i	li3i	li4i	li5i
List item (li)	Lowercase Roman (I)	li1I	li2I	li3I	li4I	li5I
List item (li)	Continuing paragraph (p)	li1p	li2p	li3p	li4p	li5p
Blockquote (bq)		bq1	bq2	bq3	bq4	bq5
Definition List	Term (dt)	dt1	dt2	dt3	dt4	dt5
Definition List	Description (dd)	dd1	dd2	dd3	dd4	dd5

It was not intended that users have to type these or even select them from a drop-down, rather they would use a add-in interfaces which aided them. The first generation is described in the XML.com article I wrote. This used a hierarchical menu system (also keyboard navigable) which was the same in n Word and OpenOffice.



The second generation of this interface is the ICE toolbar, which uses a set of buttons very like those in most modern editing application, but which tries to “Do the right thing” and apply styles, documented at the ICE site.



Saving as HTML

Using styles does not do a lot to help the quality of HTML exported from our two word processors out of the box but many third-party applications for creating HTML do try to use styles, for example ICE, or the commercial HTML Transit.

(I gather that ICE is no longer actively maintained by USQ – I'm using it here as an example of the kinds of interfaces that make it easier for users to apply styles than the defaults that come with their word processing packages – it is open source, so organisations who wanted to develop templates like the ones used in ICE could adopt part or all of it).

Heading numbering and document outlines

One of the key benefits of using heading styles is that they allow for automatic tables of contents, and to use an outline view of a document.

One issue that needs to be dealt with is document numbering. It is possible to attach numbering to styles so that the headings in a document are numbered. The simplest case is to map style names to numbers – but there are use-cases where documents have both numbered and non-numbered parts – and special cases such as appendices which might be sections at the same level as, say, chapters but have different numbers.

ICE (barely) manages to deal with this complexity by using a compound document approach with each chapter or appendix stored in a separate file. The ICE system was designed to be aggressively interoperable between the OpenOffice family and Word, which imposed a major limitation – OOo Writer can only tie ONE style to each numbering level in the document outline – with the added complication that recent versions do support 'outline level' as a paragraph attribute, although this is not tied to the Outline numbering feature as far as I can tell.

Lists

List structures are one of the most difficult things to deal with in word processing, for template design, HTML export and for basic usability.

- Word's lists have historically been very unstable. There are multiple ways to make lists in Word, including direct formatting, named lists outlines, anonymous list outlines and list styles, many of which are almost impossible to access in the new Ribbon interface that Word moved to in version 2007, there have also been many changes to the way Word handles lists and list styles over the years making this a very complicated topic.
- Writer's list support is close to unusable. The Open Document format which is the native file format has lists as a first-class object and has provision for a document to contain hierarchical list structures like those in HTML. The problem is that in a paragraph-based editing environment it is almost impossible for an author to understand the hierarchical structure of their lists – there are only very small cues in the interface to show you what level of list a particular item is on, for example, and the process of adding an extra paragraph into a list, without a bullet is bizarrely complicated – it is not a matter of applying formatting or styling, but a structural manipulation which is at odds with the way word processors typically work.

Interoperability is a problem – when transferring documents with or without styles between Word and Writer, lists often break, numbering is destroyed, and indenting changes. Even when using styles, when Writer saves to the `.doc` format, instead of creating word styles for lists corresponding to its internal ones it creates new ones. So, the result even of saving a Writer document and reloading it back in to Writer breaks documents.

Against this background I think it is worth describing the ICE approach to interoperability here as an illustration of the sort of thinking that is needed in a heterogeneous application environment.

In ICE there is a standard set of list style names which is implemented differently in Word and OpenOffice. Both share a set of paragraph styles with the same name, li1b for a first-level bullet list, li2n for a second level numbered list item and so on.

In Word

Each paragraph style is tied to a named list outline (not a list style), so the list styles `li1b`, `li2b` et al are attached to a single outline called `lib`. While Word has these named outlines they are difficult to access reliably – there is no way to pick one off a list, they only appear in galleries and if the one you want is not showing you cannot access it. In ICE use of these lists is entirely by macros which can repair them when they break. (And they do).

In Writer

There is a corresponding List style for each paragraph style, and when a user uses the ICE toolbar or menus to apply a paragraph style, a macro applies the relevant list style at the same time Writer has long had an option to tie a paragraph style to a list style, but it doesn't work reliably.

In both cases when things go wrong there is a macro that cycles through every paragraph in the document and re-applies each style, including making sure that is a paragraph is in li1b style it is attached to the correct list. In Word, there is a macro to reset its list formatting, rebuilding each named list outline, as Word has a tendency to do what can only be described as 'go crazy' and have all the lists in a document change formatting (I have not checked up on this in the latest version, but I have no reason to think that this has been fixed).

Saving as HTML

Saving lists as HTML is one of the worst performing areas for word processors. Their algorithms typically do a very poor job. Word 2010 still saves list items as paragraphs with formatting rather than as list structures, and the OpenOffice.org family produce non-standard, often flat-out wrong structures. The ICE approach of a full consistent set of styles means that ICE can create properly structured output, including correctly nesting block quotes and non-numbered paragraphs inside complex list structures. It does this by using the level numbers in ICE styles to work out what should be nested inside what.

In a potential new service for converting word processing content to HTML this could be extended to deal not only with a standard set of style names, but to infer structure in other situations as well , indenting being one of the major cues (that seems obvious, but the current algorithms in word processors and in browser based editors manage to get it wrong – they produce odd structures that are almost certainly not what any author was trying to mean).

Metadata

I looked at metadata in a blog post.

Embedding images

By default in Word and OpenOffice, if you paste in or create an image or other inline object such as a chart or drawing it 'floats' relative to the content. The idea is that objects can be placed on a page. For web and ebook publishing this is not useful and it leads to lots of frustrations. Unless very fine grained support for image placement is required for print publication it is usually best to anchor images as characters rather than as floating objects.

- Anchor images and objects as characters.

  In Writer:

  - Right click on an embedded object and choose `Anchor`, `As Character`.

  In Word:

  Right click on the object and choose `Wrap Text`, `Inline with Text`
- Use the in-built vector drawing packages for diagramming, but:
  - Don't draw on the document as though it were paper, insert objects that contain drawings.
    - In Writer: From the `Insert` menu choose, `Object`, `OLE Object`, (Name of application) `Drawing`
    - In Word: From the Insert Tab choose Shapes, New Drawing Canvas.
- Use the inbuilt Maths editors in either platform.

Maths

Maths support on the web has been a problem, but things are slowly improving. The ideal is to use MathML which is part of HTML. Current practice on the web often involves the use of LaTeX as a source for mathematics which is then rendered into HTML via other tools. There are commercial plugins for both Word and Writer that can deal with LaTeX markup.

Word 2007 and 2010 and Writer call export MathML and save MathML inside their file formats, although this does not happen when you save as HTML, so it should be possible to automate production of high quality output to HTML given the resources. As far as I know, nobody has done this yet.

For casual use of maths, using the approach I describe below of generating images using the Word processor's inbuilt Save as HTML, which creates images of the maths is probably adequate but is far from ideal where mathematics is a key part of the content.

Converting to images to HTML

One of the areas where many HTML conversion projects fall down is images. Because office suites have tight integration with drawing and presentation applications, and inbuilt maths rendering etc it is often very difficulty for external code to render anything but a plain-text document or with images that already in web formats such as JPEG or PNG as HTML from a word processor file The ICE application uses OpenOffice to render inline objects from both Word and Writer documents, and in parallel created HTML from the XML source files.

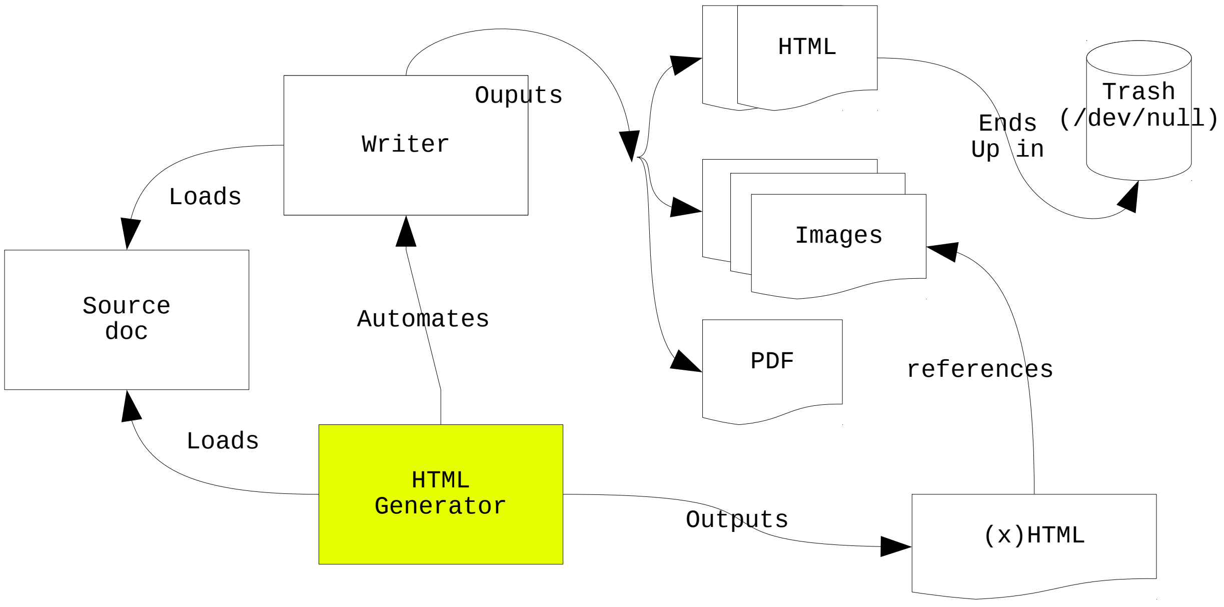


Illustration 1: Diagram showing how an HTML converter can use the word processor to create web ready images, while still creating HTML from the XML inside its native document format (.docx or .odt)

In a previous project I worked on with some members of the ICE team we simply used the HTML output from Word 2000 and massaged it to be much better quality HTML, discarding the formatting that Word outputs and using the style names (which are output as classes) to generate the HTML.

This is an important area because the various integration features which allow authors to embed charts and vector graphics and so on are one of the main reasons to keep using word processors. If candidates are working on theses that are exclusively text, then using a tool-chain such as asciiidoc with a wiki text format or Pandoc may be worth considering.

Reference managers

There is no space here for a full evaluation of reference managers such as EndNote, Zotero or Mendeley, all of which have integration with word processors, for now candidates must be assisted in choosing an appropriate tool for their discipline and institution. Regarding the future, JISC is investing in this area with support for the Open Bibliography project – one important dimension of this will be working out how cost and effort across the entire sector can be reduced by simplifying and rationalising the process of citing works. If we have a large scale open bibliography available, then referencing in many disciplines could be as simple as linking to a URI for a resource in that shared bibliography – with all the details of presenting citations and reference lists handled automatically.

Tables of contents, figures etc

Both Word and Writer have extensive automation features for tables of contents and tables of figures etc as demonstrated in the Edinburgh template. It is important to set up examples and encourage candidates to use them. A template should have the required tables of contents for headings, figures etc already in place with examples and instructions on how to insert figures etc so they are numbered. Most of these should probably be discarded in exported HTML and EPUB versions and appropriate native HTML versions prepared automatically by software.

Summary

Any new template design process needs to consider all of he above (and more) in multiple cycles, until a stable set of design constraints emerges.

- Interoperability requirements. The range of packages you want users to be able to work with imposes constraints on which features can be used. Current trends such as tablet computing, and the rise of vertical plaftorms such as Apple's iOSX devices need to be given consideration.

  (On the ICE project, several years ago we decided to support OpenOffice.org Writer and Microsoft Word to ensure cross-platform coverage across Windows, Mac and Linux – today's environment is very different, but during the ICE project our each-way bet paid off when Microsoft dropped support for Visual Basic scripting in the Mac version of office – we were able to keep coverage for the style toolbar on that platform by offering Writer to Mac users.)
- Whether to support single-file theses or multi file theses or both. Multiple files will increase the need to provide support, and possibly require the use of external tools, but for modern research theses, the ability to aggregate different things such as data files together is attractive.
- A set of styles and/or other guidelines.
- How to make HTML and EPUB versions of the content. If an application can produce HTML then that can be converted to EPUB automatically. It is producing HTML of sufficient quality that is a problem.

The ICE system I have continually referenced throughout this document was one fully worked example of all of the above considerations. It was not designed for theses, although it was tested on them and found to be adequate. But ICE is several years old, so re-doing this process now would produce a different design. Some of the key insights from the ICE design process include:

- Templates need to be immediately useful to their users. That is, people have to be able to see the point of what they are being asked to do/fill-in. For theses this is simpler than for some other types of document, the institution can say to a candidate: “Use this!” or, “Your thesis must meet the formatting criteria we specify, here is a template that helps”.
- Following from the above point, rapid feedback is required – if the final deliverable is expected to be an ebook, amongst other formats, make sure there is a system in place to show the candidate and their

supervisor to

- The document authoring system needs to be integrated into institutional processes, so making the authoring system part of the supervisor/candidate conversation, and automating submission will be important.

While this document has looked at some design issues for templates, it does not provide a solution to the question it is trying to answer; how to set up an environment for creating EPUB theses from word processing source files. I will produce one final blog post for this project outlining some potential solutions to some of the issues raised in this document as a guide to where JISC might or might not like to invest in future work.

[This document is a re-post of one for the jiscPUB project – please comment over there: http://jiscpub.blogs.edina.ac.uk/2011/08/03/template-design-issues-for-wordprocessors-and-possible-future-epub-export]