4th International Digital Curation Conference

December 2008

Embedding metadata and other semantics in word processing documents

Peter Sefton,

Australian Digital Futures Institute, University of Southern Queensland

Ian Barnes,

Digital Resource Services Program, Division of Information,

The Australian National University

Ron Ward

Australian Digital Futures Institute, University of Southern Queensland

Jim Downing,

The Unilever Centre for Molecular Science Informatics, University of Cambridge

July 2008

Abstract

This paper describes a technique for embedding document metadata, and potentially other semantic references inline in word processing documents, which the authors have implemented with the help of a software development team. Several assumptions underly the approach; It must be available across computing platforms and work with both Microsoft Word (because of its user base) and OpenOffice.org (because of its free availability). Further the application needs to be acceptable to and usable by users, so the initial implementation covers only small number of features, which will only be extended after user-testing.

Within these constraints the system provides a mechanism for encoding not only simple metadata, but for inferring hierarchical relationships between metadata elements from a 'flat' word processing file.

The paper includes links to open source code implementing the techniques as part of a broader suite of tools for academic writing. This addresses tools and software, semantic web and data curation, integrating curation into research workflows and will provide a platform for integrating work on ontologies, vocabularies and folksonomies into word processing tools.

[The work here can/will be demonstrated in a presentation if the paper is accepted]

NOTES:

Page limit is 12 pages – currently 12

Introduction

This paper briefly outlines a number of practical methods for embedding metadata and inline semantics into word processing documents, many of which have been implemented as part of the Integrated Content Environment (ICE). We do not discuss the general structure of documents using headings and so on here, but this work builds on work on capturing generic document structure and transforming word processing documents into a variety of formats, undertaken as part of the ICE project (P Sefton 2006) . ICE allows authors to write academic material of all kinds including courseware, theses (Peter Sefton 2007) and research papers, published as HTML, PDF and directly into institutional repositories [NOTE: some of this integration work is incomplete but will be done by the conference].

Being able to embed metadata and semantics in documents is important for preservation services, but will also be key to implementing true eResearch services where dissemination of research is by semantically rich documents along the lines of the Datument (P. Murray-Rust & H. S. Rzepa 2004). But more importantly for users, the techniques we describe here are designed to assist in getting work done, by making it easier to submit their work to repositories, and to allow for semantically rich publishing options that have not been available to academics working with general purpose tools.

We have specified some requirements in the interests of being able to implement these techniques immediately in the ICE content management system and as part of the TheOREM-ICE project which is going to prototype a departmental thesis management system which can be used to create and disseminate semantically rich theses (Neil Jacobs 2008).

1. The Open Document Format ODF (OASIS 2005) is the primary file format for both applications, with Writer serving as a conversion bridge for Word documents.
2. All techniques must be Microsoft Word / OpenOffice.org Writer compatible and interoperate between those packages. This precludes the use of XML extensions to Word documents which do not interoperate with Writer and the OpenDocument Format.
3. The work is designed to be metadata-format agnostic, with an initial focus on being able to provide metadata in RDF as part of an OAI-ORE (IONSREPORT 2008) description for a content item, if necessary with ad-hoc predicates. The assumption here is that different services will be able to serialize the RDF into appropriate metadata schemas.
4. The goal is to produce tools and procedures usable by general academic authors, not to create a tool that can only be used in certain contexts.

Previous work

There has been a lot of work extolling the general benefits of semantically-rich documents, particularly in the world of XML and before that SGML. For example from the point of view of the repository community, *The Case for Explicit Knowledge in Documents* (Carr et al. 2004) looks at how and why one might structure semantically useful documents, but evidently not for *word processor* users – that term is not used in their paper and the mundane word processor as an authoring tool is not addressed.

One set of projects that do address the word processing user is summarized here:

> There are several environments that aim at enhancing office-like

> environments with support for semantic annotation, for example, Semantic Word (Tallis, 2003), OntoOffice (Ontoprise, 2003), and WiCKOffice (Carr et al., 2004a,b). Semantic Word and OntoOffice provide extensions to MS Word that support semantic annotations and semantic support for document authoring. (Eriksson 2007)

However, none of the above-mentioned approaches meet our requirement for something that can be used **now** across different packages. Wickoffice (Carr et al. 2004) does not appear to be available for use, Semantic Word is an ambitious semantics web project but with heavy Microsoft Word integration that is not interoperable with OpenOffice.org Writer and not suitable for general purpose document-writing (Tallis n.d.) while OntoOffice is likewise Word-only and appears to be unavailable from the vendor's website.

We have not found any formal description of embedding semantics in word processing documents using easy to implement protocols that do not get in the way of general-purpose authoring and that meets our specified requirements, hence this paper.

General semantics versus metadata

The line between document metadata and the semantics of a document is somewhat fuzzy. For example authorship is typically regarded as metadata, while marking up the name of a chemical compound might be considered document semantics. But if we are to extract a list of compounds the then it might be considered metadata to assert "This document **mentions** H_2O" which is not too far from the notion of a document subject.

We will discuss various techniques for embedding metadata on one hand and other semantics such as the names of chemical compounds on the other, without laboring the distinction. This paper is not an exhaustive list of possible ways to mark up metadata, rather it is a snapshot of the practical methods being used in the ICE system based on several years of experience with word processing driven content managements systems. But while these methods are being added to a mature system many of the techniques listed here are implemented and not yet not user-tested. We plan to report our results in future, particularly as part of the TheOREM-ICE project.

There are three general cases under consideration with the first receiving the most attention:

1. Specific semantic terms such as chemical terms or metadata items like titles or author names.
2. Block elements such as embedded activities in educational material or side-bars in technical manuals.
3. Embedded items such as data visualizations.

References such as bibliographic citations are a special case we are not considering here.

Implementation considerations

The text below notes which of the embedding techniques are implemented[1].

[1] At the time of writing the code is available in the following modules in the ICE system. Check out revision 9835 of http://ice.usq.edu.au/svn/ice/trunk using Subversion (code may be moved in future).
xhtml-export/ooo2xhtml/ooo2xhtml.py (line 245)
xhtml-export/ooo2xhtml/ooo2xhtml_states.py (line 155)
xhtml-export/ooo2xhtml/ooo2xhtml_basic_states.py (line 103 & line 187)

General techniques for structuring word processing documents

This section takes a brief look at some of the mechanisms that can be used to add structure or semantics to word processing documents in the context of our assumption.

Word processing documents have structure at two levels:

1. **The structure of the underlying file format.** This includes the basic units, such as paragraphs, higher level structure divisions, 'sections', tables and text containers (frames) as well as mechanisms for numbering items such as figures or tables and collating tables of contents for them. In addition to these structural units, both Microsoft Word and ODF have the concept of a document 'outline' which is implied, rather than marked up directly in nested XML. Generally speaking the underlying structure of the document is of little use in this project.

   Microsoft Word allows users to embed arbitrary XML into documents, but this is (a) not interoperable with Writer and the Open Document Format standard, and (b) involves a lot of configuration and programming to create usable interfaces.
2. **An 'implied' structure at higher level via styles**. While the underlying structure of word processing files has a reasonably flat structure, without nested document sections, that is not to say that word processing documents necessarily lack structure. By default in both Word and Writer there are styles for headings; `Heading 1`, `Heading 2`, etc. and these can be used to divide the document into sections so the word processor and other programs can compile hierarchical views of the document, such as a table of contents from the styles.

From within the options above there are a number of ways of adding inline information. The most obvious would be to use fields, but this creates problems with Word / Writer interoperability. For example as of mid 2008, Writer's Word import drops the data component of fields altogether. To get reliable interoperability our experience and prototyping has show that the best techniques are:

1. Styles, to record semantics in-text.
2. Tables to mark sub-structures and create form-like interfaces for user to fill out.

Styles

Style interoperability between Word and Writer is generally good. Even though there are some formatting differences both character (inline) and paragraph style names will be preserved on import and export between Word and Writer. This means that if a style is used to express a semantic element, such as an inline style `given-name`, the style marker will interchange between word processors even if there are minor differences in rendering. There are some caveats around list-styles, but these are not generally relevant to metadata work.

Experience with style-based systems has taught us that users have trouble picking styles off long lists, so we have previously created hierarchical menus for applying styles by function (Peter Sefton 2005). Following from this work the authors and collaborators at he University of Southern Queensland created a new interface device that attempts to reproduce the kind of toolbar that appears in most word processing applications, with buttons for changing the structural function of paragraphs and spans of text. That work will be the subject of more research articles, but a demonstration is available at the ICE project website[2]. More work is required to make this toolbar user-

[2]http://ice.usq.edu.au/presentations/demos/html_from_ooo.htm

extensible.

But for many of the uses outlined here users would not be expected to apply styles at all – merely to replace sample text in a supplied layout with their own metadata.

Tables

As with styles, tables are implemented in a reasonably interchangeable way between Word and Writer making them a good way to introduce structure into a document. For metadata, a document template could have a pre-created table into which users can type metadata, or for a more flexible solution, fragments of metadata can be stored in autotext (pre-loaded text modules that can be inserted in a document), applied by customized buttons or menus.

Generally speaking tables are useful for microformats that do not have to be inline with other text.

Microformats

The techniques for embedding metadata in documents we describe here use a mixture of styles and tables; essentially they are word processor based microformats. Khare and Çelik describe microformats in the context of XHTML:

> Microformats are a clever adaptation of semantic XHTML that makes it easier to publish, index, and extract semi-structured information such as tags, calendar entries, contact information, and reviews on the Web. This makes it a pragmatic path towards achieving the vision set forth for the Semantic Web. (Khare & Çelik 2006)

Pragmatic aptly describes our approach, too; as we have to work within the limitations of not one but a number of existing software solutions, standards and formats.

Implementation details

ICE's internal metadata and semantics model

In the examples below we show how metadata and other semantics can be encoded in a document. Internally, the ICE system processes this information in a number of ways.

1. For metadata it maintains an internal hierarchical data structure seen in an example below. The metadata can be exposed as MODS or as RDF, for example as part of a an OAI-ORE resource map.
2. Some semantic markup is simply passed through to an HTML rendition, for example geographical metadata is left in the HTML rendition, and can be used by downstream scripts to generate maps.
3. A plugin system can be used to extend ICE so that it can recognize certain kinds of meaningful objects and process them appropriately for print and online renditions.

Document properties for metadata

The most obvious place to store metadata is in the document properties, accessible via interfaces in major word processors. There are, however severe limitations to this approach, the most important one being that interoperability between Word and Writer is extremely limited. Limitations include that document properties don't allow for storing a formatted abstract, Writer doesn't allow for changing/setting the author and in neither word processor is there a way to associate multiple authors each with their own affiliation and email address.

Document properties are also limited to flat name-value metadata which is inadequate for describing relationships such as a document's authors and the authors' affiliations. So, we are left with ways to embed metadata within the document text itself, much as the organizers of this conference/journal have done with their template[3].

Metadata schemas

The first question in designing an embedded metadata encoding scheme is should we indicate the metadata namespace in the encoding protocol? For example, if we decided to have a metadata field for author's name, would it be `meta-author` or `meta-mods-author`?

We opted for the former: to minimize the length of style names; to minimize the confusion that authors might experience if they see unfamiliar strings like 'MODS' or 'DC' (for Dublin Core); to allow for re-mapping to other metadata schemas; and to allow for metadata that is common in papers but not catered for properly in a given schema (for example there is no field for an author's email address in MODS but it is very commonly included in research articles).

This implementation is open for review, and does not preclude adding a namespace indicator in future.

Embedded objects

One of the simplest kinds of semantic embedding happens when the object being embedded is a discrete whole. The best example of this is equations, where software solutions exist to allow a user to enter an equation which can be extracted as MathML or LaTeX (LaTeX3 2001). The built in equation editors in Word and OpenOffice.org interoperate using OLE and third-party software such as MathType (Wikipedia contributors 2008) can be used to extract equation semantics.

We are taking this approach with chemistry, by finding items that have been embedded using the proprietary ChemDraw software package and processing them with open source software to attempt to extract semantics. ChemDraw does not inherently include semantics so these need to be inferred, a process that is not reliable. This is implemented as ICE plugin code[4].

A lighter weight approach also already implemented in ICE is to link text or an image to an external file, such as a Chemical Markup Language (CML) (Peter Murray-Rust & Henry S Rzepa 2003) file. For a print paper, the text or image is left in place, but for an online version an appropriate visualization can be supplied, in this case using the Jmol (Jmol contributors n.d.) applet. A demonstration[5] is available on the ICE website.

Using tables

As discussed above, one reliable and interoperable method for adding structure to a document is to use a table, as a container for a microformat.

[3]Template for the Digital Curation conference: http://www.dcc.ac.uk/events/dcc-2008/template/
[4]Chemdraw to CML conversion code: http://ice.usq.edu.au/svn/ice/trunk/apps/ice/plugins/ice-functions/plugin_cdx2cml_function.py
[5]http://ice.usq.edu.au/presentations/demos/cml_ice_ice.htm

The simplest example is a two-column table. In this example no special styles are used, but the header-row indicates that the table is a metadata table by convention only. In the left column will be the names of the metadata items, and in the right column the corresponding values.

Document information	
Author Name	Ian Barnes
Author Affiliation	ANU

Example 1: A simplest possible table for metadata, recognized entirely by convention (not implemented in ICE).

This method for embedding metadata is a reasonable approach as it is very easy to implement. Experience has shown that simple tables like this can work as authors are unlikely to change text such as ‘document information’ particularly if they know it is important to a computer system.

A minor refinement is to use a style for the table header so it need not read ‘Document Information’. This is illustrated in the table below with the style name shown in curly braces:

Metadata {meta-document-information}	
Title	Metadata in ICE documents
Author Name	Ian Barnes
Author Affiliation	ANU
Author Email	Ian.Barnes@anu.edu.au

Example 2: A refinement on the simple metadata table where the table is indicated with a style name in the header row (implemented in ICE)

With multiple authors, each author’s extra data must follow their name in document order so that the software can sort out which affiliation and email address belongs with which author. This means that for multiple authors from the same place the affiliation will have to be entered multiple times. The text in the left column is used to create a hierarchical data structure.

```
{ 'title':['Metadata in ICE documents'],
 'author':[{'name':'Ian Barnes',
            'affiliation':'ANU'},
           {'name':'Peter Sefton',
            'affiliation':'USQ'}
          ]
}
Example 3: An example data structure of metadata extracted from
a table (implemented in ICE) note that each item is an array
which could contain more than one value, for example multiple
authors.
```

There is a potential refinement to this table-based method using more complex table structures, but we have not been able to devise an easy-to interpret table layout that would make it clear that a number of authors share an affiliation. Here is an example of a structure that shows that name and affiliation are both sub-parts of the author metadata.

Document information		
Title		Metadata in ICE documents
Author	Name	Ian Barnes
	Affiliation	ANU

Example 4: An unworkably complex table for showing metadata hierarchies. (not implemented in ICE)

Metadata tables are appropriate for some types of document, and many reports and forms already have a similar structure. It is possible to more-or-less hide the metadata using a macro to hide text, and set table borders to be blank – the method varies slightly between MS Word and Writer.

Another use of tables is to use them as bounding boxes for blocks of content. For example the ICE system uses them for embedding slides. See this document on the ICE website[6]. Using buttons to the top-right of the page, the document can be re-rendered as a slide presentation. The microformat is very simple, slides are marked by a table, which contains at least one paragraph in `h-slide` style. Users can create slides around key parts of a document without disrupting document flow by formatting them with no borders, or choose to format them to stand out from the text. A support person can set up blank slides as autotext so that users can drop them in to document with a few keystrokes or mouse clicks. The same approach is used in ICE for embedding educational content such as activities or lists of readings.

Encode metadata name in paragraph style

One of the most flexible methods of embedding semantics in a document is to use styles. Styles can either apply to a paragraph, 'paragraph styles' or to a span of characters 'character styles'. The approach we have taken is to define styles such as `p-meta-title`, `p-meta-author-name`, `p-meta-author-affiliation`, `p-meta-author-email`, `p-meta-abstract`, `p-meta-keywords`, and use these to mark up the metadata.

The algorithm to process metadata still needs to see metadata in document order to be able to associate authors and their email addresses and affiliations, for example, but this is easy to achieve using either tables or by adding sections to a document formatted as multiple columns.

Metadata in ICE documents {style: p-meta-title}
Ian Barnes {style: p-meta-author-name}
ANU {style: p-meta-author-affiliation}
Ian.Barnes@anu.edu.au {style: p-meta-author-email}
Peter Sefton {style: p-meta-author-name}
USQ {style: p-meta-author-affiliation}
sefton@usq.edu.au {style: p-meta-author-email}
Abstract: This is the abstract. Does this belong here, or is this mechanism unsuitable for abstracts, especially since they can have multiple paragraphs? {style: p-meta-abstract}
This is the second paragraph of the abstract. {style: p-meta-abstract}
Keywords: metadata, ICE, word-processing {style: p-meta-keywords}

Example 5: Encoding metadata using paragraph styles (styles are shown in curly braces) (implemented in ICE).

[6]http://ice.usq.edu.au/introduction/about.htm

As with the metadata table method this method has no way of distinguishing between given name and family name. But see below, where we can add inline styles to split names into parts.

One complicating factor with this approach is if a header like “Keywords:” is included at the start of the keywords paragraph, then there has to be special code to detect and remove it. A better approach is to use a style that inserts the header text automatically.

As the amount of metadata grows, so will the number of styles. If users are expected to apply the styles from a list using the built-in features of their word processor this would be unsustainable but we expect that in most cases pro-forma templates will be provided where users change sample text rather than having to understand the styling system themselves.

Styles for general semantics

ICE has a general-purpose method for embedding arbitrary semantics via a convention where one can extend existing styles. For example see this blog post[7] describing a preliminary exploration of adding geographical semantics to documents using a very simple convention, hinging on the definition of a new style i-geo for use in marking up text inline.

We are working on extending this to include other domains, including chemistry. One approach will be to run the [8] chemical semantics engine over a document as part of The-OREM while it still being authored, getting it to mark-up semantics that it has identified. This contrasts with the current approach where the tool is usually usedon published material, where there is no opportunity to check that it has identified the correct items. One complication is that the same text could potential refer to different or several chemical entities (e.g. "glucose", refers to a family of sugars and “ice” could be a content management system, frozen water or a dangerous drug). The author will be able to add things that OSCAR misses or correct it and mark parts of the document such as acknowledgments using need a style such as `i-chem-ignore` or `p-chem-ignore`.

One unresolved issue is how to handle references to molecules in-text where the molecule is depicted/described in an embedded object, but needs to be referenced.

User-extensible metadata

One simple approach that allows user to add ad-hoc metadata to their documents if to use the paragraph style `p-meta`, and then have the name of the metadata as the first word of the paragraph content, followed by a colon.

```
Abstract: This is the abstract.  {style: p-meta}
And more abstract. {style: p-meta}
And yet another paragraph of the abstract. {style: p-meta}
Keywords: metadata, ICE, word-processing {style: p-meta}
```

Example 6: A general purpose metadata style where the user can specify the sub-type of metadata using a header followed by a colon. (Implemented in ICE)

[7]http://ptsefton.com/2008/06/19/adventures-in-geocoding-part-2-embedding-data-points-in-documents.htm
[8]OSCAR Toolkit: http://www.rsc.org/Publishing/ReSourCe/AuthorGuidelines/AuthoringTools/ExperimentalDataChecker/getOSCAR.asp

Metadata headings

Here we have a special paragraph style to indicate that a heading marks the start of a metadata item. The system then keeps adding items to the indicated metadata field until it hits the next heading or the next metadata item. This helps to separate the heading for a metadata item from the actual metadata content, for example in the abstract.

> Abstract {style: h-meta-abstract}
> This is the abstract. Does this belong here, or is this mechanism unsuitable for abstracts, especially since they can have multiple paragraphs. {style: p}
> This is the second paragraph of the abstract. {style: p}
> Keywords: metadata, ICE, word-processing {style: p-meta-keywords}

Example 7: Marking metadata using a heading style to indicate a block of metadata (implemented in ICE).

Inline styles for metadata hierarchies

To get good metadata, we sometimes need to go below the paragraph level, for example to separate out the family name and given name when we specify a person's name. To encode this information in a word processing document, we either need separate table cells in method 1 above, or we need special inline styles for marking up the name parts. We use inline styles `i-meta-familyname` and `i-meta-givenname` for this.

In each case here, the paragraph has paragraph style `p-meta-author-name`, the given name has character style `i-meta-givenname` and the family name has character style `i-meta-familyname`.

> {inline-style: i-meta-givenname Ian} {inline-style: i-meta-familyname Barnes} {style: p-meta-author-name}
> ANU {style: p-meta-author-affiliation}
> Ian.Barnes@anu.edu.au {style: p-meta-author-email}

Example 8: More delicate metadata using inline styles (implemented in ICE)

So the resulting data structure is as follows:

```
{'author':[{'name':'Ian Barnes',
            'givenname':'Ian',
            'familyname':'Barnes',
            'affiliation':'ANU'}]
}
```

Example 9: More detail for an author in the ICE-internal data structure.

Dissemination

ICE can be programmed to serialize its internal metadata structure in a variety of formats. One format that is already implemented is MODS:

```
<mods:mods xmlns:mods="http://www.loc.gov/mods/v3">
  <mods:titleInfo>
    <mods:title>Metadata in ICE documents</mods:title>
  </mods:titleInfo>
  <mods:name type="personal">
    <mods:namePart type="given">Ian</mods:namePart>
    <mods:namePart type="family">Barnes</mods:namePart>
    <mods:displayForm>Ian Barnes</mods:displayForm>
    <mods:role>
      <mods:roleTerm type="text">author</mods:roleTerm>
    </mods:role>
    <mods:affiliation>ANU</mods:affiliation>
  </mods:name>
...
```

Example 10: ICE serialization of metadata.

In the course of the TheOREM-ICE project we will be integrating ICE content with repositories via the OAI-ORE protocol – as part of this work, document metadata and embedded chemical semantics will be exposed as part of an ORE resource map, so that chemical theses can be published as part of the semantic web.

Conclusions

In some sense this paper is stating the obvious. It catalogues some techniques which have been widely used in document templates but which do not seem to have been the subject of formal research.

The approach we have taken is to list and implement a wide variety of ways that metadata and other semantics can be embedded in real documents, either by an end-user or by a support person. This flexibility should allow the widest possible range of existing templates and practices to be adapted. We have avoided techniques which would lock users into a particular software environment or standard, focussing instead on interoperability. The ICE system contains open source code, written in Python, which implements these methods. ICE can be used as a web service, or the code could be re-used in other systems which have GPL-compatible licensing.

One possible use for this is to streamline publishing and archiving: as the metadata is already encoded in the document, ICE can pre-fill the various forms associated with sending a document for publication or submitting it to a repository for archiving, extending previous work on creating a scholarly workbench application (Barnes 2006) and on repository integration with workflow tools (Monus et al. 2008) (Pearce et al. 2008) in the Australian repository community.

Acknowledgements

Portions of the work reported here sponsored by the Australian Commonwealth Department of Education, Science and Training, (DEST) under the Systemic Infrastructure Initiative (SII) as part of the Commonwealth Government's Backing Australia's Ability - An Innovation Action Plan for the Future (BAA).

Thanks to Peter Murray-Rust and Joe Townsend at the The Unilever Centre for Molecular Science Informatics for input into this document and to Linda Octalina at the University of Southern Queensland for implementation.

References

[article-journal] Barnes, I., (2006). Integrating the Repository with Academic Workflow. *OpenReposiotries. Sydney, APSR http://www. apsr. edu. au/Open_Repositories_2006/ian_barnes. pdf.*

[paper-conference] Carr, L. et al., (2004). The case for explicit knowledge in documents. In *Proceedings of the 2004 ACM symposium on Document* *engineering*. Milwaukee, Wisconsin, USA: ACM, pp. 90-98. Retrieved February 22, 2008, from http://portal.acm.org/citation.cfm?id=1030417

[article-journal] Eriksson, H., (2007 ,July). The semantic-document approach to combining documents and ontologies. *International Journal of Human-Computer Studies*, 65(7), 624-639. Retrieved February 22, 2008, from http://www.sciencedirect.com/science/article/B6WGR-4NFXDN3-1/2/b2dfe578e6aa84b801e80a84b7888a35

[article-journal] IONSREPORT, S., (2008). OAI-ORE specifications. *Scholarly Communications Report*, 12(1), 5-5.

[book] Jmol contributors, *Jmol: an open-source Java viewer for chemical structures in 3D*, Retrieved July 23, 2008, from http://jmol.sourceforge.net/

[paper-conference] Khare, R. & Çelik, T., (2006). Microformats: a pragmatic path to the semantic web. In *Proceedings of the 15th international conference on World Wide Web*. Edinburgh, Scotland: ACM, pp. 865-866. Retrieved February 25, 2008, from http://portal.acm.org/citation.cfm?id=1135777.1135917

[webpage] LaTeX3, P., (2001). LaTeX2E for authors. Retrieved , from http://www.latex-project.org/guides/usrguide.pdf

[webpage] Monus, L. et al., (2008 ,April). Zero Click Ingest. Retrieved May 20, 2008, from http://pubs.or08.ecs.soton.ac.uk/119/

[article-journal] Murray-Rust, P. & Rzepa, H.S., (2004). The Next Big Thing: From Hypermedia to Datuments. *Journal of Digital Information*, 5(1), 248. Retrieved , from http://jodi.tamu.edu/Articles/v05/i01/Murray-Rust/?printable=1

[article-journal] Murray-Rust, P. & Rzepa, H.S., (2003). Chemical markup, XML, and the World Wide Web. 4. CML schema. *Journal of Chemical Information and Computer Sciences*, 43(3), 757-72.

[webpage] Neil Jacobs, (2008). Departmental Thesis Management System development using the Integrated Content Environment (TheOREM-ICE). Retrieved July 14, 2008, from http://www.jisc.ac.uk/whatwedo/programmes/digitalrepositories2007/theorem-ice.aspx

[article] OASIS, (2005). OpenDocument v1.0 specification. Retrieved , from http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf

[article-journal] Pearce, J. et al., (2008). The Australian METS Profile–A Journey about Metadata. *D-Lib Magazine*, 14(3/4), 1082-9873. Retrieved , from http://www.dlib.org/dlib/march08/pearce/03pearce.html

[webpage] Sefton, P., (2006). The Integrated Content Environment for Research and Scholarship. *ICE Website*. Retrieved April 30, 2007, from http://ice.usq.edu.au/introduction/ice_rs.htm

[paper-conference] Sefton, P., (2007). An integrated approach to preparing, publishing, presenting and preserving theses. In *ETD 2007*. Uppsala. Retrieved July 2, 2007, from http://eprints.usq.edu.au/archive/00002653/

[article-magazine] Sefton, P., (2005). XML.com: Hacking Open Office. *XML.com*. Retrieved February 25, 2008, from http://www.xml.com/pub/a/2005/01/26/hacking-ooo.html

[article-journal] Tallis, M., Semantic Word Processing for ContentAuthors.

[chapter] Wikipedia contributors, (2008 ,July). MathType. In *Wikipedia, The Free Encyclopedia*. Wikimedia Foundation. Retrieved July 23, 2008, from http://en.wikipedia.org/w/index.php?title=MathType&oldid=225118687