Editing Marks for XML Markup

Table of Contents

1. Editing Marks for XML Mark-up - Abstract

Editing (or proof-reading) marks have been used as a shorthand in copy-editing and proof-reading tasks since the 15th Century (André, 1998), and in a similar form to the current International Standard since the 17th Century (Simpson, 1935). The marks are designed for highlighting spelling, typographical, composition and grammatical errors, where the document being reviewed is in the same presentational form in which it will be published.

When documents are created in XML-based semantic mark-up languages such as DITA and DocBook, the content is separated from the presentational form. Copy-editors and proof-readers work not with the final form of the document, but with a *form-agnostic* representation of the document marked up with semantic identifiers. The final form is produced in an automated publishing process, where semantic mark-up is mapped to presentational elements. When working with a form-agnostic document, the editing and proof-reading tasks focus on language issues (spelling, punctuation, grammar, and wording errors) and on semantic mark-up issues.

Editing in XML-based semantic mark-up languages calls for a different approach to editing marks. Addressing this problem is the main motivation for this paper. This paper suggests an alternative set of editing marks for semantic authoring, incorporating those conventional editing marks that remain appropriate, and discarding those marks that are not.

A further impediment to editing in XML-based semantic languages is the challenge of displaying the XML mark-up in paper-based renditions of the document without significantly degrading the readability. This paper identifies two technical approaches to address this challenge.

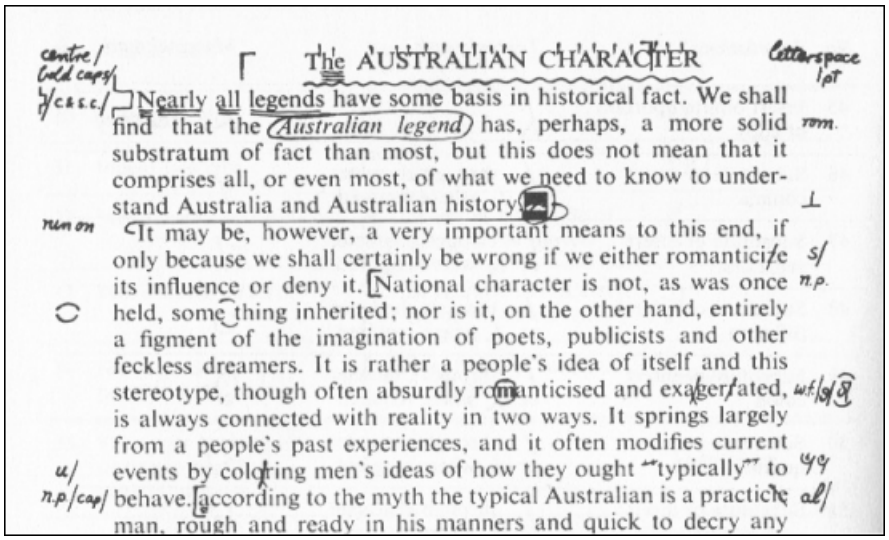
2. Existing Use of Editing Marks

Editing is an often tedious process in which a document is scrutinised to improve the quality of two fundamental components: content and form.

Editing marks (also known as *proof-reading marks*) are a hand-written mark-up system used by editors of all many of documents. They are commonly used in technical publications. An understanding of the marks is widely known. Most generic style manuals such as the Australian Style Manual: for authors, editors and printers (Snooks and Co., 2002) provide a guide to the use of editing marks. (Refer to Figure 1).

Some common editing marks indicate changes such as insert, delete, transpose, set in capitals, italicise, bold, spelling, close space, align vertically, move left, and new paragraph. Many of these marks relate to presentational style, and few can be used to indicate errors or required changes in semantic mark-up. (Refer to Figure 2).

Figure 1. Example of Proofing Marks in Practice



Editing marks have been in use since at least the 15th Century, according to André (1998) in Petite histoire des signes de correction typographique. In the English-speaking world, according to Simpson (1935), a system of editing or proofing marks was being used in printing shops in the 17th Century. As the printing industry grew, editing marks became more widespread and standardised.

André and Richy (1999) explained the purpose of editing (or proof-reading) marks as to precisely localize where to modify the text, and to precisely indicate where modifications had to be done.

According to the Freelance Editorial Association (n.d.), there are four different categories of editing:

- developmental (editing during the content creation process)
- substantive (improvements after the writer has completed the manuscript)
- copy (correcting errors and enforcing style manual rules)
- proofreading (checking the final version for typographical and layout errors).

The Chicago Manual of Style (1982) divides the editorial process into *mechanical editing* and *substantive editing*. Mechanical editing involves such matters as capitalisation, spelling, agreement of subjects and verbs, punctuation, and use of numbers. Substantive editing involves re-writing, re-organising, and reworking of the writing style to conform with the guidelines in a style manual.

Regardless of the categorisation, editing marks are used in each of the stages of editing.

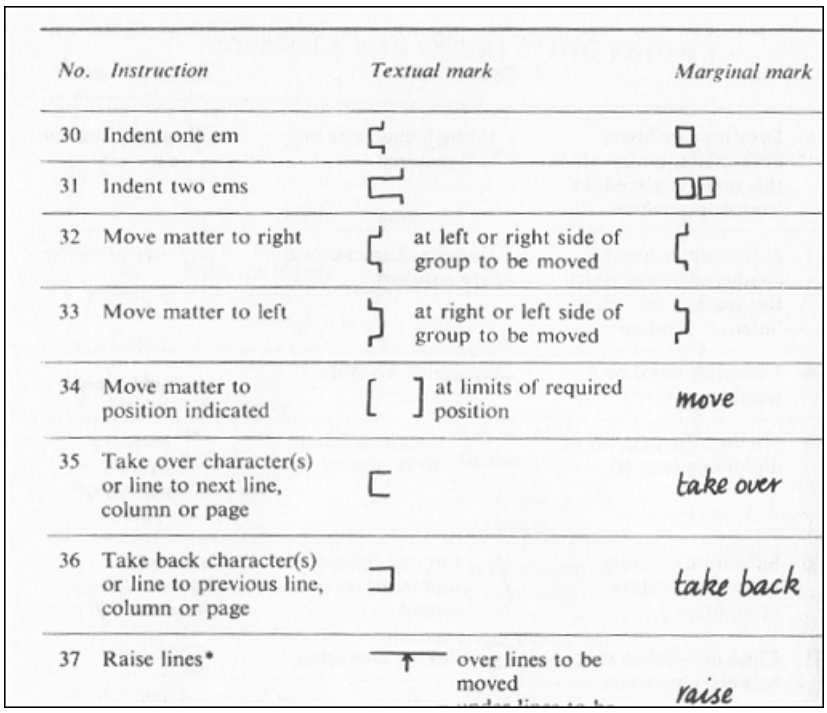
In this paper, editing is used to describe the whole editing process, and proof-reading is used to describe the *mechanical editing* checks of the final version of a document prior to publication.

The eradication of grammatical and spelling errors is important to reinforce the quality, credibility and integrity of the document; if it is printed with spelling mistakes and editorial and design inconsistencies, its authority will - to some extent - be undermined (Snooks, 2002, p. 276). It is therefore imperative that adequate and efficient proof-reading measures are taken for all documents.

Although computerisation has improved some of the processes, editing remains a labour-intensive endeavour. As explained in Snooks (2002):

> With the introduction of computer technology, the proofreading process is sometimes reduced to little more than running a spellcheck. This is not proofreading, and it is far from adequate. ... Proofreaders must be able to find and correct errors, understand copy editors' markings and indicate corrections in the proper way.

Figure 2. Sample proof mark guidelines from Manual of Style (Snooks, 2002)



No.	Instruction	Textual mark	Marginal mark
30	Indent one em		
31	Indent two ems		
32	Move matter to right	at left or right side of group to be moved	
33	Move matter to left	at right or left side of group to be moved	
34	Move matter to position indicated	at limits of required position	move
35	Take over character(s) or line to next line, column or page		take over
36	Take back character(s) or line to previous line, column or page		take back
37	Raise lines*	over lines to be moved	raise

Editing marks therefore remain an important tool in the editing process. An international standard for editing marks, ISO 5776: Graphic technology - Symbols for text correction, was released in 1983 to formalise long-standing industry conventions. However, while the standard editing marks are useful for correcting traditional documents, they are not effective for correcting XML-based, semantic documents such as those created in DITA.

3. The Need for New Editing Marks for XML Mark-up

The use of semantic mark-up in DITA, where text elements are marked up based on their meaning, allows the content to be completely separated from its rendition and display to the reader. For example, a term is marked up as a `<term>` and a citation as a `<cite>`, and no information about how those elements will be displayed is stored in the content. Stylistic (display) rules are applied when the DITA content is *transformed* into a reading format, such as HTML or paper. In a DITA workflow, documents are created as collections of modular, re-usable topic files, and mechanisms allow not only the format to be separated from the content, but also the context. The same topic may be a section in the context of one publication, but a sub-section in the context of another. The intermingling of content, format and context in a style-based document workflow essentially eliminates the possibility of re-use. Once a paragraph is styled as having a 13 cm left margin, it cannot be used on paper 12 cm wide. A phrase marked up in italic won't render as italic on a reading device that doesn't support italic. But a citation identified as a citation in a DITA topic can be processed to italic by one transformation process, to bold red by a different transformation process, and to synthesised voice by another transformation process.

Traditional style manuals for editors and publishers have a significant emphasis on presentational aspects of documents. Self (2009, p.258) found that up to 70% of the content of popular style manuals, such as The Chicago Manual of Style (1982), was devoted to issues related to presentational style, or form. In semantic mark-up environments, the author, editor and proof-reader have no association with form, as form is separated from content to become a discrete, separate, automated function within the document production process.

Documents written in a semantic mark-up format such as DITA cannot be proofed in the same way as documents written in a presentational format. The semantic mark-up doesn't show the eventual presented form of the document; when printed or displayed for proof-reading, it shows a possible presentational rendition. This is sometimes known as *WYSIOO*, or *What You See Is (just) One Option* (O'Keefe, 2006). The author, editor and the proof-reader may never see the presentational form of the document until after publication. The different publishing workflow in a semantic mark-up environment is required, and a different approach to editing, proof-reading and editing marks

is required. Gardiner (2010) suggested that one of the reasons why the promised efficiencies of XML-based publishing hadn't been achieved was because it is obvious that the [editing] workflow continues parallel the conventional stages of production.

Studies as far back as 1987 have demonstrated that when proof-reading on-screen instead of on paper, more proof-reading errors were missed, fewer pages were read, and there was a greater accumulation of fatigue during the reading session (Wilkinson, 1987). Wilkinson (ibid.) concluded that material be printed for proof-reading. O'Hara & Sellen (1997) found that one of the major advantages that paper offers in supporting annotation while reading. Editing and proof-reading XML documents should not preclude the use of paper in the process, so if editing or proof-reading on paper is to continue as part of the publishing process, a system of editing marks has to be maintained. Research into paper-less editing and proof-reading, such as that by André and Richy (1999), tend to rely on electronic pens, tablets and paper. When such paper-less approaches reach maturity, the need for an *XML-friendly* editing mark system (and a method for exposing underlying XML mark-up) will remain.

4. Personal Observations - Editing an XML Document

One of the authors of this paper has first-hand experience editing and proof-reading documents in DITA XML format, and made a number of observations about the limitations of conventional editing marks for this process.

The author's personal observations were as follows.

> When editing a document written collaboratively in DITA (the DITA Help Technologies Guide), my first step was to print a copy of the generated PDF version of the DITA document. I find it more convenient to edit on paper. I found, however, that the standard editing marks were not appropriate for this task.
>
> The problems that arose are typified by the following examples.
>
> - If a phrase was rendered in italics when I was expecting bold, how would I know if that was a mark-up error (such as the author using `synph` when `uicontrol` was semantically correct) or a processing decision? How do I indicate that I want to check this or stipulate this? My role was to ensure the DITA document was accurate, not to adjudicate on the presentational style.
> - In some cases, a semantic mark-up did not result in a different presentational style from standard text. How could I check that the element was correctly marked-up if there was no visible indication?
> - Topic metadata, and even some topic content, may not be rendered in the printed version. How can I check whether this important information is correctly included?
> - If a quotation appeared on the page correctly encapsulated with quotation marks, did this mean that the author had correctly used the `q` element, or had incorrectly hard-coded the quotation marks?
> - If I could determine that a chunk of text was incorrectly marked-up, how could I indicate the error with editing marks?

In summary, the author found that editing marks were inadequate because:

- they were primarily focussed on presentational appearance
- they did not allow for revising semantic mark-up

For editing of paper documents authored in DITA to be practical, the following changes to editing practice are needed.

- A new output format specifically used for editing, where all semantic mark-up is identifiable in a supra-organisational standard way, must be devised.

- A new standard editing mark-up scheme for printed renditions of semantically marked-up documents must be developed.

5. Editing Marks Proposal

A new set of XML-aware editing marks to replace or augment the current style-based editing marks will permit handwritten instructions to be communicated between authors, reviewers and editors, if used in conjunction with a technique to expose the semantic mark-up in the review document.

This paper proposes the following editing marks for reviewing XML documents.

Table 1. New XML-specific editing marks

Intention	In-text mark-up	In-margin mark-up
Apply the element in the margin to the highlighted text (or check that it has been applied).	( ) or circle	<cite> or <cite>?
Apply the attribute in the margin to the highlighted element (or check that it has been applied).	or ( ) or circle	@product=Lite
Insert a link at the marked insertion point to the topic (or other element) name nominated in the margin.		*[topic name]*
Remove the XML mark-up	( ) or circle	<>
Move the opening XML tag to the position marked by the arrow		<>
Move the closing XML tag to the position marked by the arrow		<>

Table 2. Traditional editing marks to continue to use with same meaning

Intention	Mark-up
Insert in text the content indicated in the margin	
Delete	
Delete and close up	
Leave as printed	*stet*
Change to capital letters	*caps*
Change to capital for initial letter and small caps for remainder	*c. & s.c.*
Change to lower case	*l.c.*
Insert superior (superscript) character(s)	
Insert inferior (subscript) character(s)	
Use ligature or dipthong	
Close up (delete space)	
Insert space	#
Transpose characters or words	*trs*

Intention	Mark-up
Begin new paragraph	
No fresh paragraph; run on from previous paragraph	
Spell out abbreviation or figure in full	
Insert omitted portion of text	
Substitute or insert a comma	
Substitute or insert a semi-colon	
Substitute or insert a full stop	
Substitute or insert a colon	
Substitute or insert a question mark	
Substitute or insert an exclamation mark	
Surround with parentheses	
Surround with brackets	
Insert an apostrophe	
Insert an ellipsis	
Insert a diagonal stroke	
Refer issue... doubtful accuracy	
Correction is concluded (separator for in-margin mark-up)	

Table 3. Traditional editing marks not to use

Intention	Mark-up
Change highlighted text to italics	
Insert more white space	
Change highlighted text to small capitals	
Change highlighted text to bold	
Change highlighted text to roman (non-italic) type	
Wrong font	
Invert type	
Change damaged character	
Underline	
Insert space between lines or paragraphs	
Reduce space between lines or paragraphs	
Make space appear equal between words	
Less space between words	
Insert letter space	

Intention	Mark-up
Move to centre	⌜⌝*centre*
Indent text	⊂
Move text block to the right	⊂
Move text block to the left	⊃
Move text to a different position	[]*move*
Take (wrap) text over to the next line	⌊*take over*
Take text back to the previous line	⌋*take back*
Raise lines	*raise*
Lower lines	*lower*
Correct the vertical alignment	‖
Straighten lines	═
Push space down	⊥
Insert a rule (en, em or 2 em)	⊢
Insert a single quotation mark	‘’
Insert a double quotation mark	“”
Insert a leader	⊙

6. Editing Aids to Expose XML Mark-up

The observations of the impediments to editing XML-based semantic documents on paper, using editing marks, highlighted the need to expose the XML semantic mark-up of a document to the editor or proof-reader. One of the authors of this paper has experimented with a number of technical solutions based on DHTML, CSS and XSL-T to allow the display of XML mark-up in paper-based renditions of the document, without significantly degrading the readability. Two such technical solutions can be best explained through brief case studies.

Case Study: Editing-Friendly Web Content through DHTML

The structured nature of XML documents opens up many opportunities for semi-automating some editing and proof-reading tasks, and providing editing aids.

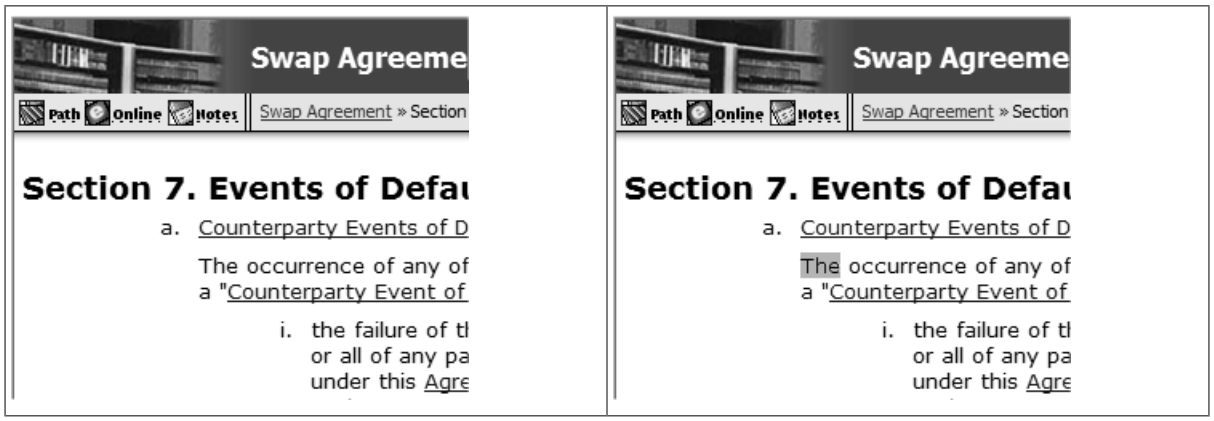
A company producing electronic versions of complex legal contract documents required a quality assurance (QA) process that would effectively eliminate the risk of errors being introduced into documents during the conversion from paper-based (Microsoft Word or Adobe PDF format) to electronic (Microsoft HTML Help, or *CHM*, format). The accuracy of the electronic documents was fundamental to the business.

There was no alternative to a manual proof-reading and QA process to ensure that the automatic conversion steps had been completed correctly. This manual process was tedious and repetitive, and thus prone to human error. For example, the proof-reader was required to check that every use of a defined term (such as Project Completion Date) in the document was correctly linked to the definition of that term in the Definitions section. In some cases, similar terms could easily be linked to the incorrect definition, such as Commonwealth (Bank) being linked to Commonwealth (Government).

To make the QA process easier for the proof-readers, one of the authors of this paper developed a DHTML (Dynamic HTML) feature which would highlight the elements in the document of interest to proof-readers. For example,

unlinked words starting with a capital letter (a possible indicator or a defined term) might be highlighted so that the proof-reader could check whether the word was a term or an undefined proper noun or first word in a sentence (as shown in Figure 3).

Figure 3. Screen captures showing different rendering views



Two different renderings of the CHM document was required: the normal view, and the QA view. To minimise the risk of the QA version being accidentally distributed to end users of the document, and to obviate the need to build two different versions of the document, the DHTML technique used employed an external JavaScript file named `qa.js`. When an HTML topic in the CHM file was displayed, a JavaScript function embedded in the topic was launched to attempt to load the `qa.js` file. If the file was not found, the topic would be displayed normally. If the file was found, functions in the `qa.js` would execute to re-format the topic to highlight the point of interest. Different `qa.js` files could be used to highlight different points of interest for the different QA sweeps. As end users would never have the `qa.js` file, they would never see the QA view.

The JavaScript code to load the `qa.js` file was:

```
<script>
if(location.href.search(/QA/)!=-1)  {
 document.write('<scr' + 'ipt order="6" language="JavaScript1.2" src="' + chmfile_path + 'qa.js">
 }
</script>
```

While this technique was very effective for on-screen proof-reading, it was not used for paper-based editing. However, it would be possible to print the topics to provide a paper equivalent provided the formatting was suitable for the (usually monochrome) paper medium.

Case Study: Exposing Mark-up through CSS

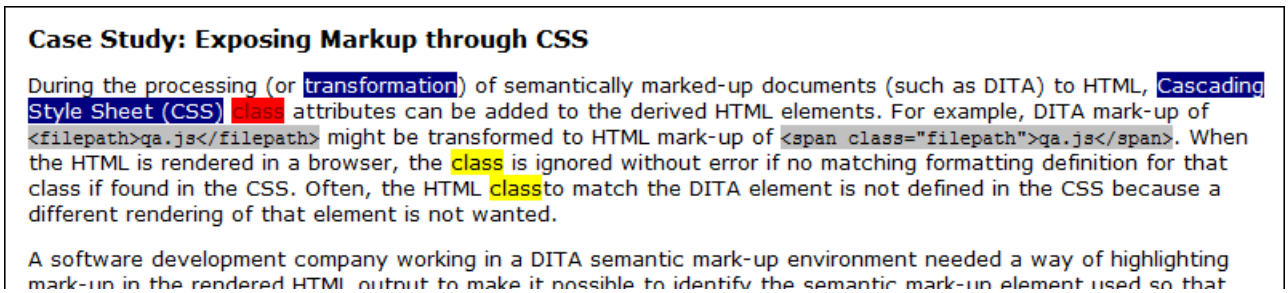
During the processing (or *transformation*) of semantically marked-up documents (such as DITA) to HTML, *Cascading Style Sheet (CSS)* `class` attributes can be added to the derived HTML elements. For example, DITA mark-up of `<filepath>abc.exe</filepath>` might be transformed to HTML mark-up of `<span class="filepath">abc.exe</span>`. When the HTML is rendered in a browser, the class is ignored without error if no matching formatting definition for that class if found in the CSS. Often, the HTML class to match the DITA element is not defined in the CSS because a different rendering of that element is not wanted.

A software development company working in a DITA semantic mark-up environment needed a way of highlighting mark-up in the rendered HTML output to make it possible to identify the semantic mark-up element used so that consistency of mark-up could be checked as part of the editing process. One of the authors of this paper worked in conjunction with the company to create a CSS file which would highlight the mark-up using a colour-coding schema. The approach developed required two versions of the HTML document to be produced: one for the editing and checking purpose, and one for display to the end user. However, it would be possible to also use this technique in conjunction with the DHTML technique described earlier.

The colour-coding simply applies different colours and/or fonts to different mark-up elements, such as `synph` DITA elements displayed in maroon. (Refer to Figure 4.)

An expansion of the technique currently under development is to move away from simple colour coding (which does not suit printing to monochrome media, and has accessibility problems) to richer, exposed mark-up. The expanded technique takes advantage of CSS2 features such as `:before` and `:after` *pseudo elements*. In some cases, CSS is proving to be too limited, and modifying the XSL-T templates that control the processing from DITA to HTML is a more comprehensive approach.

Figure 4. Example of CSS used for colour-coding the origin elements of processed HTML

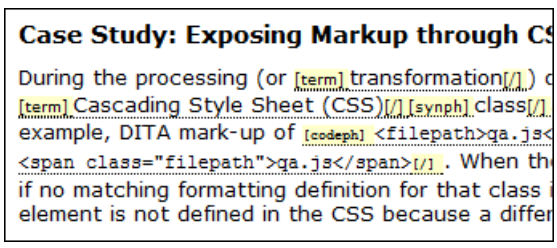


The CSS additions to colour-code the output HTML based on its semantic origin would follow the pattern:

```
.synph {
 color:maroon;
 background-color: red;
}

.term {
 background-color: navy;
 color:white;
}

pre.codeblock, samp.codeph {
 font-size: 120%;
 background-color:gray;
}

span {
 background-color: yellow;
}
```

Using the `:before` and `:after` *pseudo elements*, the output can show the semantic origin in a more sophisticated way, with the principal benefit being that semantic labelling can be printed to monochrome media. (Refer to Figure 5.)

Figure 5. Example of CSS pseudo elements used for exposing the origin elements of processed HTML



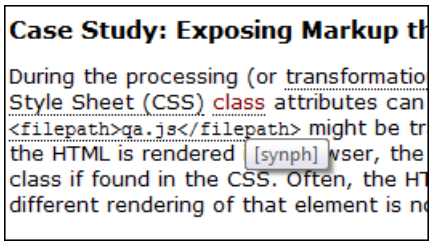
The CSS additions to prefix the original DITA element name would follow the pattern:

```
.term {
 background-color: white;
 color:black;
 border-bottom: #000 1px dotted;
}
```

```
.term:before{
 content:"[term]";
 background-color: yellow;
 font-size: 70%;
}
.term:after{
 content:"[/] ";
 background-color: yellow;
 font-size: 70%;
}
```

Changing the XSL-T transformation templates can make it possible to show the semantic origin without cluttering up the text with mark-up. An HTML `acronym` element (with a `title` attribute value of the name of the original DITA element) can be injected around the `span` element normally added. This `acronym` element results in a *hover text* or *tooltip* pop-up displaying the semantic origin when the user holds the mouse over the element (Self, n.d.). A visual indicator that text has mark-up is discrete, dotted underlining. (Refer to Figure 6.) This technique is only useful for on-screen reading, as the pop-ups do not appear on paper!

Figure 6. Example of `acronym` tag used for identifying the origin elements of processed HTML



The supporting CSS would follow the pattern:

```
acronym {
 cursor: help;
 border-bottom: #000 1px dotted;
}
span {
 color: maroon;
 background-color: yellow;
}
```

Using techniques such as these make it possible to print a rendition of the document with exposed semantic mark-up, in turn making it possible to edit and proof-read XML-based semantic documents on paper.

7. Editing Marks for XML Mark-up - Conclusion

The standard editing marks used to indicate required changes in a printed document are not suitable for semantic authoring, where different presentational forms of a document can be automatically produced from the same source. Changes to editing practice are required to make editing of semantic documents more effective.

This paper proposes two approaches to make it practical to edit XML-based semantic documents on paper:

- Use an alternative set of editing marks for semantic authoring, incorporating those conventional editing marks that remain appropriate, and discarding those marks that are not.
- Use technical solutions based on DHTML, CSS and XSL-T to allow the display of XML mark-up in paper-based renditions of the document without significantly degrading the readability.

The adoption of these two approaches will together allow paper-based editing and proof-reading of documents to continue into the era of XML-based documents that separate content from form.

8. Editing Marks References

- André, J. (1998). Petite histoire des signes de correction typographique. Cahiers GUTenberg, (31), 45-59.
- André, J., & Richy, H. (1999). Paper-Less Editing and Proofreading of Electronic Documents. EuroTeX'99 Proceedings. Retrieved July 8, 2010, from http://www.irisa.fr/imadoc/articles/1999/heidelberg.pdf.
- Gardiner, D. (2010, May). Discovering XML for editing. The Canberra Editor, 19(4), 6-10.
- ISO 5776: 1983: Graphic technology - Symbols for text correction. (1983). International Organization for Standardization.
- O'Hara, K., & Sellen, A. (1997). A Comparison of Reading Paper and On-Line Documents (pp. 335-342). Presented at the Proceedings of CHI '97, Human Factors in Computing Systems, Atlanta. Retrieved July 13, 2010, from http://scholar.google.com.ezproxy.lib.swin.edu.au/scholar?q=kenton+o%27hara&hl=en&lr=&btnG=Search.
- O'Keefe, S. (2006, April 12). Wednesday at WritersUA: Structured authoring: taking the plunge. Scriptorium. Retrieved May 15, 2010, from http://www.scriptorium.com/blog/2006/04/wednesday-at-writersua-structured-authoring-taking-the-plunge.html
- Snooks and Co. (2002). Style manual: for authors, editors and printers (6th ed.). Brisbane: Wiley Australia.
- Self, T. (n.d.). The Mysterious Acronym Tag. HyperWrite. Commercial, . Retrieved July 6, 2010, from http://www.hyperwrite.com/Articles/acronym_tag.aspx.
- Self, T. (2009). DITA and the Challenges of Single-Source Article Publishing. In *Communication, Creativity and Global Citizenship: Refereed Proceedings of the Australian and New Zealand Communications Association Annual Conference*. Presented at the ANZCA Annual Conference, Brisbane. Retrieved July 2, 2010, from http://www.cpe.qut.edu.au/conferences/2009/anzca/proceedings/Self_ANZCA09.pdf.
- Simpson, P. (1935). Proof-Reading in the Sixteenth, Seventeenth and Eighteenth Centuries (1st ed.). London: Oxford University Press.
- The Chicago Manual of Style. (1982). (13th ed.). Chicago: University of Chicago Press.
- Wilkinson, R. T. (1987). Proof-reading: VDU and paper text compared for speed, accuracy and fatigue. Behaviour and Information Technology, 6(2), 125-133. doi: 10.1080/01449298708901822.