E-MELD Consultant's Report 2004

What a Linguist Needs to Know about Word Processing

or: Don't hack it - design it!

Dafydd Gibbon

Universität Bielefeld, 2004-10-01 (V03 - draft, comments welcome)

1. Objective

This report explains why and how to use stylesheets in word processing, from a linguistic perspective. The present document is used as a model.

2. Motivation

Many users of word processors are unfamiliar with the advantages of using well-defined styles for document objects like titles, headings, paragraphs of specific types, which have the same consistently defined rendering properties (font, spacing, centring, indentation etc.) wherever they occur in a document. A common practice is to assign rendering properties of this kind to each local occurrence of a document object separately: a stretch of text is marked, and instantly formatted. For short documents this is a handy, "quick and dirty" procedure. For long and complex documents it leads to waste of time, patience and energy (and of course the personal satisfaction of having produced a professionally formatted document is missed!).

The advantages of using styles are:

1. Time saving: if the rendering properties have to be changed for each individual occurrence of a document object such as a heading, this is unnecessary time-wasting - change a style once to change all occurrences of this style at the same time.
2. Consistency preserving: if the rendering properties are changed separately by hand, inevitably errors occur, leading to inconsistency of formatting.
3. Consistent cross-media conversion: well-defined styles in one formatting system, such as a word-processor, can be converted consistently into well-defined styles in another, such as the world-wide web.
4. Consistent cross system conversion: with well-defined styles, there is a much greater chance of comparable results when different word processors or different versions of a word processor are used.
5. Support of barrier free access: screen readers for the visually impaired can process styles intelligently and produce well-structured output.

Some examples:

- wherever headings of the same level occur, they are expected to look the same, so why not assign the format once and for all;
- wherever text paragraphs occur, they are expected to look the same as the others of the same type;
- wherever bibliographical entry paragraphs occur, they are expected to look the same as the others of the same type, and so on;
- wherever indentations occur, they should be of the same size;
- wherever spaces before and after paragraphs occur, they should be consistent, and also freely variable.

In each of these cases, why not assign these properties once and for all, rather than

separately, each of the many times that they occur in a document? If well-defined styles are not used, none of these advantages can be used.

3. A linguistic perspective on document production

Ordinary word processing - as used in linguistic articles and reports (including this one) - from the applied linguistic perspective of *text generation*, not from the naive vantage point of *writing with a computer*. The idea is that linguists should not march from phonemes through morphemes and words to sentences and stop there, but should regard texts as systematic language objects which are worthy of their systematic attention, and apply this knowlede professionally in their own writing. These language objects are made of document objects with semantic properties and media properties, just like other language objects with semantic properties and media properties (e.g. phonetic interpretations), can be described with a document grammar, and receive text-structural descriptions, for example as tree-diagrams. Essentially, then, the report is about practical aspects of styles and stylesheets from the applied linguistic point of view.

The moral of the story is that if linguists (particularly linguists concerned with language documentation, corpus tagging, XML markup and the like) do not use these objects and their properties intelligently in generating their own texts with a word processor ... well, maybe the moral should not be made too explicit, after all. This document, by the way, was composed using these techniques with OpenOffice (version 1.1.), and will be distributed both in the OpenOffice writer format (sxw; compressed XML) and in OpenOffice-generated PDF format, following the open source, open archive philosophy (see also http://www.openoffice.org).

4. Applications of linguistics in word processing

Evidently, *words* are within the purview of linguistics, and it is not surprising that linguistics is heavily involved in the development of word processors like OpenOffice or MS-Word. A moment's reflection will show that the following properties of words, dealt with by modern word processors, are actually the domain of linguists, which is why linguists are employed by software companies to solve issues concerning such properties:

- Spelling (cf. spell checkers).
- Correct inflection (cf. grammar checkers).
- Thesaurus as a writer's help.
- Word completion facility.
- Capitalisation facility.
- Use of correct quotation marks.
- Translation of terms for localisation to other languages.

There are also many add-ons to word processors concerned with functionalities such as the reading aloud of selected text portions, translation of selected text portions, which involve the results of many years of research in linguistics, phonetics, speech technology and text technology. There are, of course, imperfections in each of these applications: nobody is perfect, no theory is complete, and neither are paper dictionaries and grammars.

5. Beyond word processing: documents and document objects

The term "word processing" is something of a misnomer. No user of a word processor presumably just wants to write words. The goal is usually to write a *document*, consisting of *sections*, consisting of different *paragraph* types such as *headings*, and *text*, each paragraph consisting of *lines*, and each line composed of a stream of *characters*. Note

that this account has nothing to say about linguistic units such as words and sentences. In fact, this is the level of *Applied Text Linguistics*, in which *texts* are dealt with in much the same way as *sentences* are dealt with in more conventional linguistics.

The objects dealt with in Applied Text Linguistics are, among others, those just introduced. For convenience of reference they will be called *DocumentObjects*. There are many kinds of Document Object, but the main Document Objects handled by word processors are:

1. document,
2. page
3. paragraph,
4. character,
5. list (enumerating list or bullet list),
6. table.

The basic Document Objects are document, page, paragraph and character; the other two types are more specialised.

6. Document objects and their properties

Objects of all kinds have specific collections of properties, and Document Objects are no exception. The properties which Document Objects have are of two main types:

1. structural properties, which define the components of the Document Object and the contexts into which it fits (e.g. for paragraph objects: what the next paragraph object should be),
2. interpretative properties, which define what *meaning*, i.e. *content* a Document Object has (this is entirely up to the author and the readers, of course) and what *rendering* style a Document Object has (i.e. what its appearance will be).

The relation between document structure on the one hand, and its two interpretations as *content* and *rendering* on the other, are shown in Figure 1.

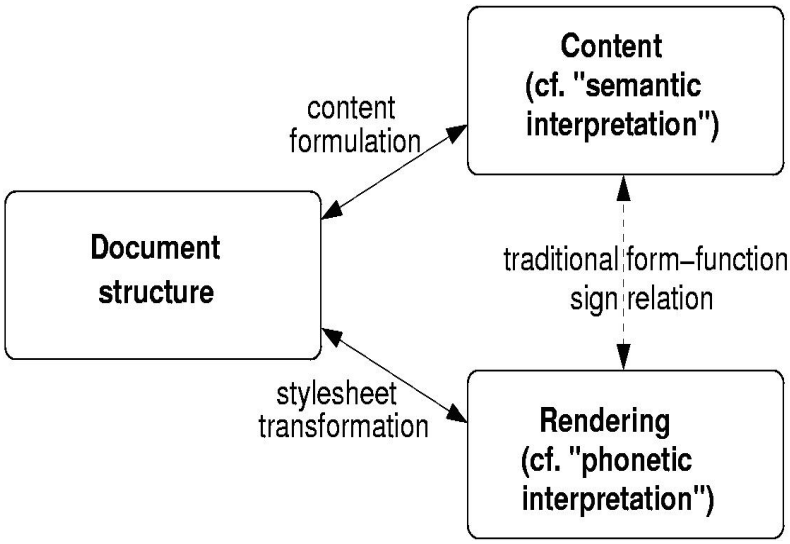

Figure 1: Text linguistic view of Document structure, Content and Rendering.

The properties are typically organised as *attributes*, i.e. collections of mutually exclusive properties, which are referred to as the *values* of the attributes. For example, a paragraph

Document Object cannot be both centred and left-aligned at the same time, and a character Document Object cannot be both 12pt and 10pt, or italic and plain at the same time. Examples of some Document Objects and some of their attributes and values are given in Table 1.

Note that the four Document Objects in Table 1 are not in a fixed hierarchy, but in a partial order: pages do not necessarily correspond to paragraphs, and paragraphs do not necessarily correspond to pages. Pages and paragraphs belong to two orthogonal - independent - levels of organisation. Documents consist of paragraphs, and paragraphs consist of characters. But simultaneously, documents consist of pages, and pages consist of columns (one, two or more). The difference can easily be characterised:

1. The paragraph is a part of *document structure*, which organises and structures the meanings intended by the author. The paragraph also has *rendering properties*, which reflect the intentions of the author with respect to the paragraph's status as title, heading, subheading, etc., and of course *content* properties.
2. The page has no significance with respect to the intentions of the author (in general; bibliophile books are the exception which proves the rule), but reflects production contraints determined by the publisher.

Table 1: Document Objects and some of their properties

Document Objects	*Attribute*	*Typical values*
Document	Name	relevant metadata
	Author	relevant metadata
	Version	relevant metadata
Page	Size	sub-attributes: size, top, bottom, left, right margins, header, footer, frame, ...
	Margin	sub-attributes: top, bottom, left, right margin
	Header	special page-linked paragraph type
	Footer	special page-linked paragraph type
Paragraph	Alignment	left, right, centred, justified
	Numbering	enumerated, bulleted
	Tabulator	horizontal tab settings
	Indentation	left & right margin, first line indentation
	Spacing	gap above and below
	Line	1, 1.5, 2, ...
	Frame	sub-attributes: line type, thickness, colour, ...
Character	Font	Arial, Helvetica, Times Roman, Courier, ...
	Size	10pt, 12pt, ...
	Bold	bold, non-bold
	Italic	italit, non-italic
	Underline	underline, non-underline
	Colour	red, orange, ... white, black

7. Styles

The definitions of rendering properties for Document Objects are known as *styles*, and a set of such styles for a document is known as a *stylesheet*. Stylesheets are generally provided by publishers for books, scientific articles, and so on, and traditionally are printed on paper, with examples. However, in electronically based publishing the stylesheets are simply the definitions of Document Objects in a word processor. Using the formatting facility of the word processor, a set of styles can be defined and assigned to an entire document stylesheet. For OpenOffice, this is can be stored as a template file with the conventional ".stw" ending (rather than ".sxw"), and in MS-Word as a template file with the ".dot" ending (rather than ".doc").

8. Special paragraph object types

The concept of paragraph object used in word processors is very general, and many different subtypes of paragraph object can be defined. The subtypes inherit the basic rendering properties defined for the standard paragraph type; these properties may be overridden by the specific properties needed for the paragraph subtype. The paragraph subtypes used in the present document are:

1. Pre-Title (user-defined)
2. Title (predefined but modified),
3. Subtitle (predefined but modified),
4. Author (user-defined),
5. Version (user-defined),
6. Heading 1 (i.e. level 1, not the first heading; predefined but modified),
7. Text body (predefined but modified),
8. Footer (predefined but modified).

Clearly, in all but plain typewriter rendering style such as those used in the body of plain emails, no distinction in terms of rendering properties occurs. However, in the context of a word processor, rendering differences can easily be defined. Some paragraph subtypes are predefined by the word processor, as indicated in the list. Many other paragraph subtypes are typically required in a long and complex document and can be defined by the author. Very useful examples of predefined styles are the paragraphs in a table of contents, which point to page numbers of sections, bibliographical references, or the paragraphs in a bibliography which contain the individual bibliographical entries.

Note that if the correct heading levels are used, a table of contents can be constructed automatically, and satisfactory renderings in other media, e.g. the web, can be achieved.

The styles are assigned to segments of text by means of *markup codes*; the most familiar rkup code system is HTML, the markup language used for web documents.

9. Are you a word processor hacker?

If you do any of the following, as a linguist you should be ashamed of yourself...

1. To create a space between paragraphs you insert an empty line (actually an empty paragraph object consisting of one line). DON'T DO THIS: define the space before and after the paragraph type which suits your needs. WHY? Preceding and trailing spaces are properties of paragraphs and should be defined in the style for the relevant paragraph type. If you just use an empty paragraph (i.e. type the return key) you are committed to a fixed spacing; if you define the spacing for each paragraph separately,

you run the risk of inconsistencies and waste time; if you define a style, you will avoid these problems.

2. To create a table you use spaces and tab separators, and draw the table using separate graphical lines over the text. DON'T DO THIS: create a table object, and assign frame properties to it. WHY? Spaces and tab separators may be defined differently on different machines even with the same word processor, and the table will consequently look a mess. A table rendered in this way is extremely hard to edit, and will not export consistently into another medium. A table object, with properly defined styles for cells, rows, columns and framing is quick to construct, easy to edit, and exports consistently.
3. To create a list, you write numbers at the left of a new paragraph. DON'T DO THIS: use a list object, either with enumeration (pick the numbering style) or with bullets (pick the bullet style). Why? If the list style does not suit you (e.g. the wrong kind of indentation is used), you can edit the style once for all and stop worrying. If the bullet points do not suit you, replace them in the style with other symbols, and the same applies.
4. To create a heading, you mark the line (actually the paragraph) and directly assign centring, font size, etc. to the marked section. DON'T DO THIS: define the rendering style properties for the Document Object. WHY? This is a common, but really not so clever practice, which is a real time-waster and inconsistency-causer: imagine that an editor requires the headings to be changed to a different font, font size, or highlight convention, and imagine how long this would take for an article or a book if each heading were changed separately. Not only that, imagine the potential for errors and inconsistencies. If you change the style, you change it once, and all headings of the same type change automatically.
5. To change the numbering of a heading or subheading, you add the number by hand to each heading individually. DON'T DO THIS: add a numbering property to the heading style once, and all headings will change automatically. This ensures automatic numbering which is both consistently counted and consistently formatted.

10. Document structure

The structure of a given document can be described using a tree-graph derived from a document grammar, just like the structure of a sentence. The structure of the present document, for example, is illustrated in Figure 2.

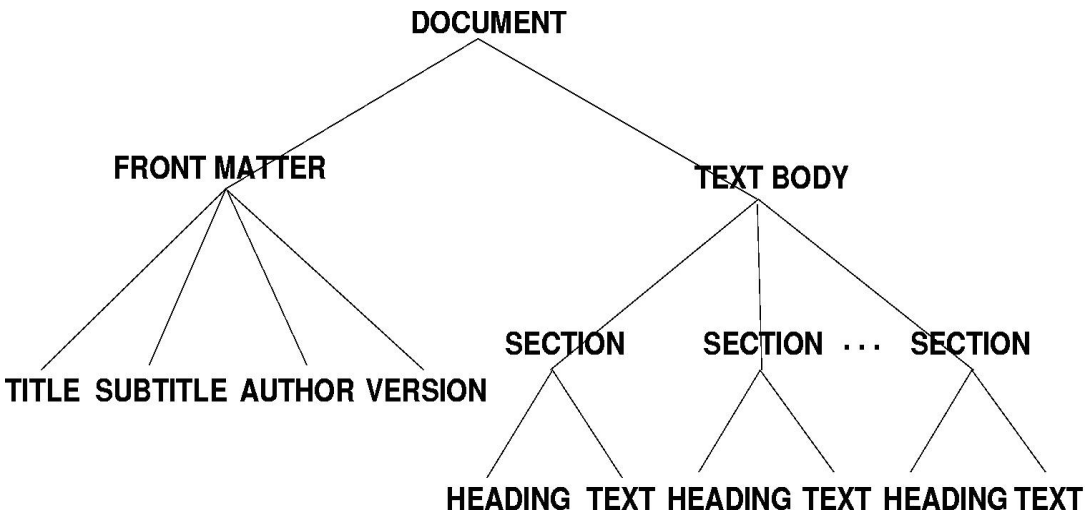

Figure 2: Document structure for the present document (simplified).

For example, a document may consist of front matter, a main text, and back matter. The front matter will include title, author, version (e.g. date, edition, table of contents, etc.), and in the case of a book an be very complex. The main text may consist of chapters, sections, subsections, subsubsections and so on, which in turn consist of Text Objects such as paragraphs, lists (in turn consisting of list elements), tables and so on. The back matter may consist of endnotes, bibliography and index. A linguist will not find it hard to write a grammar of this kind on the lines of the grammars used for the description of sentences of a language. Further details will be given below.

11. How to ...

The following instructions pertain to OpenOffice but can be transferred easily to the corresonding menu functions in commercial word processors. Only the basic features of style definition are dealt with.

1. Designing the text structure and the rendering:
    1. Draw up a list of the paragraph Document Object types (such as Title, Heading 1, Heading 2, etc.) which you need (see the list given above for the present document).
    2. Construct a table with 3 columns (for Document Objects, attributes and values).
    3. Enter each of these paragraph Document Objects in the leftmost column (cf. the table given above).
    4. Enter the attributes required for each object, and modify the default paragraph level values which these attributes should receive (e.g. spaces, justification, numbering, line centring).
    5. Also enter the attributes which all character objects in this paragraph type should receive by default, and modify the default values accordingly (e.g. font, size, italic, colour).
    6. This table is your *stylesheet*.
2. Implementing the rendering:
    1. Open a new text document.
    2. In one of the text fields in the function bar, the style field should appear. In a new document, this style entry is likely to be called "Standard", "Normal", etc. Click on the select button attached to this field, and very likely you will only see this style listed.
    3. Navigate the menu space via the "Format" menu point, then "Styles", then "Catalogue".
    4. A long list of pre-defined styles will appear.
    5. In the lower left text field select "All Styles" instead of "Automatic", and even more styles will appear.
    6. Check this list for styles in your list; stick to the names for standard predefined styles where possible.
    7. Click on any style name, and a dialogue box with around 12 tabs at the top willappear. Each tab groups related attributes together.
    8. *The default values of the attributes can then be modified according to your stylesheet.*
    9. If no suitable style is defined in your list, define a "New" style: the dialogue box with the 12 or so tabs will appear, first of all requesting a name for the file, the name of another style to switch to when the "Return/Enter" key is pressed, and a basic style from which the new style inherits its defult properties.
    10. *The default values of the attributes can then be modified according to your stylesheet.*

11. Note that no explicit distinction is made in this dialogue box between tabs for *paragraph*, *line* and *charater* properties. The distinction is rather obvious, but it will be useful to list the tabs pertaining to each type of Document Object.
12. Commercial word processors use similar techniques for style definition; the appearance of the menu points and dialogue boxes is very different, of course.

3. Test your style definitions by writing a short document containing
   1. the styles you have modified or defined, such as
      1. Title
      2. Heading 1 (i.e. top level heading)
      3. Heading 2 (i.e. next level subheading)
      4. Text body
   2. and additional Document Objects such as lists and tables, with their styles defined appropriately.
   3. Check the styles used in this report.

When you have done this, you will have

1. designed a text document on linguistic principles, and in particular
2. defined a stylesheet in the form of an object-attribute-value table containing a set of styles relevant to your document.
3. implemented the stylesheet using a word processor,
4. tested the implementation in the production of a document.

12. A practical example: the present document

Figure 2 shows a reproduction of the first page of an earlier version of the present document as a sequence of document objects, along with a stylised version of the Document Objects and the names of the rendering styles assigned to them.

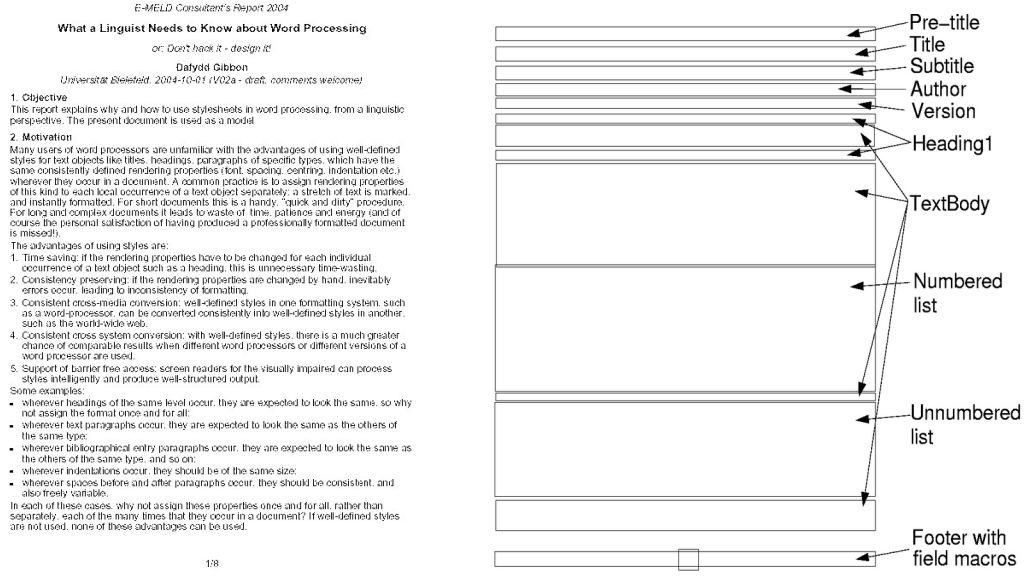

Figure 2: Document objects on p. 1 of an earlier version of the present document.

The page object itself is not labelled; it has upper, lower, left and right margin attributes, as well as a footer attribute. The stylised version of the document object shows the

bounding boxes which show the rendered size of the document objects, determined by:

1. the size of the objects it contains (e.g. for a paragraph object, the number of lines it contains and their spacing),
2. its own size rendering attributes (e.g. for a paragraph object, its left, right, upper and lower margins).

The rendering attributes assigned to each style obtain their values by inheriting them from a default style, in this case called "Standard", unless they are specifically modified to suit the special requirements of the document object concerned. For instance, the "Title" object contains one or more paragraphs with modified upper and lower spacing, which in turn contain characters with modified values for the size (16pt) and highlighting (bold) attributes. The main attribute-value assignments for the paragraph and character objects in the styles are shown in Table 2.

Table 2: Main rendering attributes and values of objects in the present document

Style	*Inherited from*	*Paragraph attributes*	*Character attributes*
Standard:	default	Left indentation: 0.00cm Right indentation: 0.00cm First line indentation: 0.00cm Upper spacing: 0.00cm Lower spacing: 0.00cm Left justified	Arial standard 12pt
Pre-title:	Standard	Upper spacing: 0.40cm Lower spacing: 0.40cm Centred	Italic
Title:	Standard	Upper spacing: 0.40cm Lower spacing: 0.40cm Centred	Bold 14pt
Subtitle:	Standard	Upper spacing: 0.40cm Lower spacing: 0.40cm Centred	Italic
Author:	Standard	Upper spacing: 0.20cm Lower spacing: 0.20cm Centred	
Version:	Standard	Centred	
Heading1:	Standard	Outline numbered Upper spacing: 0.40cm Lower spacing: 0.20cm	
TextBody:	Standard	Upper spacing: 0.07cm Lower spacing: 0.07cm	
Numbered list:	Standard	Numbering	
Unnumbered list:	Standard	Bulleted	
Footer:	Standard	Centred	10pt
Field macros:	Footer		

13. A simple grammar for the present document

The following phrase-structure grammar describes the arrangement of document objects on the first page of the current documents. In order to complete the grammar for the whole document, table and figure objects need to be added.

DOCUMENT	→	FRONT-MATTER DOCUMENT-BODY
FRONT-MATTER	→	Pre-Title Title Subtitle Author Version
TEXT-BODY	→	SECTION SECTION*
SECTION	→	Heading1 TEXT
TEXT	→	TextBody | NumberedList | UnnumberedList

There is a clear analogy to standard phrase structure /constituent structure, context-free) grammars for sentence syntax of the "S → NP VP" type. In the notation used here, the arrow "→" can be read as "consists of", the space between the categories on the right-hand side of the arrow as "followed by", the asterisk "*" as "none or arbitrarily many", and the vertical bar "|" can be read as "or" (here the exclusive or).

The notation used in different document description contexts varies in detail, but all notations employ some means of indicating hierarchical or *immediate dominance* structures meaning "consists of", alternatives, and sequence or *linear precedence*.

14. Summary

The main point to be made is that a document has a *text linguistic tructure* which is comparable with *sentence structure* in conventional linguistics, and that these receive a *rendering interpretation* which is comparable with *phonetic interpretation* in conventional linguistics. These structural levels have much in common:

- In semiotic terms, both documents and sentences are complex signs, which can be thought of as abstract or mental units, which are interpreted in terms of reality at two levels, i.e. as *meaning* and *form*,
- paradigmatically, the constituent objects and their properties are organised into an inheritance hierarchy, the most common kind in conventional linguistics being a type hierarchy, the most common kind in word processor text linguistics being a default hierarchy,
- syntagmatically, the constituent objects are organised into a compositional hierarchy, rather like the constituents of a word or a sentence,
- The same formalisms can be used for describing the structure, the meaning and the form properties of constituent objects, for example context-free grammars, or as attribute-value structures and inheritance hierarchies.

The constraints on text structure are of course different in detail from constraints on sentence structure, just as constraints on sentence structure are different from constraints on word structure. For the syntagmastic hierarchy, typically *document grammars* are defined in order to permit electronic analysis and display of documents; in the XML document markup framework, grammars are expressed as Document Type Descriptions (DTSs) or XML Schemata.

Topics of this kind are dealt with in the discipline of Text Technology and a number of related disciplines, including Computational Linguistics and Eoftware Engineering. The most widespread kind of approach goes under the heading of XML technologies, which comprise conventions for document grammars and tools for creating, checking and presenting documents systematically, and are gradually being adopted as a standard for the creation and rendering of internet documents. Of particular interest for the humanities

is the use of these techniques to classify, archive and search documents systematically, and in this area text technology is closely related to library and archive sciences. The Text Encoding Initiative (TEI) is perhaps the most well-known approach to document description using these technologies. These technologies are not bibliographically documented here because the standard references can be found on the internet by straightforward keyword searching, and in any case are often internet documents themselves.

It should be emphasised that there are many more dimensions to document production than those described here. The focus of attention in the present document was on the classic written paper. Many more rendering techniques are available, of course, including rendering as a hypertext (in graphical or text browsers), rendering by reading aloud with a speech synthesis application for blind users or users in visually adverse environments.

15. Conclusion

The moral of this story is that you, as a linguist should, design documents in a linguistically motivated fashion, practising what you preach, and bearing in mind how document structure relates to both your content and to the appropriate rendering style for each Document Object and its content. By doing this you will

- save time,
- be consistent,
- ensure the same appearance in different word processors and on different platforms,
- support barrier-free document access in assistive technologies for the disabled.

For short, informal documents of temporary relevance, a quick and dirty solution with no well-defined Document Objects, and using local format assignment of paragraph and character properties may be fine. For long or complex documents, advantage should be taken of style definitions.

And of course you do not have to be a linguist to understandthese points or produce professional stylesheet-based documents.