PART II

Text Technologies

5. Markup Technology and Textual Scholarship

CLAUS HUITFELDT

This chapter gives a brief overview of the background and development of markup systems—that is, formal languages for the representation of electronic documents.[1] The chapter focuses on aspects of markup technology that are particularly relevant to textual scholarship. It gives an introduction to some of the key concepts of the Extensible Markup Language (XML) and the Text Encoding Initiative (TEI) and considers some of their limitations, possibilities, and future potential.

Introduction

About This Chapter

This chapter is intended as an introduction to markup for textual scholars who are new to the subject. In order to assess the relevance or utility of markup to activities within textual scholarship, there is no need to be conversant with all aspects and details of the technology. What is required, however, is a grasp of its basic presuppositions, possibilities and limitations, and the investments and risks involved. Therefore, most of what is covered here is of a quite general nature, though focusing on issues assumed to be of particular relevance for textual scholarship. Much that would be appropriate for a wider or different audience has been left out, and little will be of interest for readers of this chapter who are already familiar with the basics of markup.

Why Textual Scholars Should Care about Markup

According to one usage of the word "markup," all documents are marked up. In this view both punctuation, page layout, and typography are examples of

markup. Consequently, the reason why textual scholars should care about markup is simply that it is present in all documents, whether in the form of electronic or printed documents, manuscripts, or other written documents.[2]

In a more specific usage of the word, "markup" refers to the use of special markers in electronic document files in order to signal certain properties of the document. However, even according to this more restricted usage of the word "markup," it may firmly be maintained that virtually all electronic documents are marked up. And since textual scholarship increasingly has to relate to electronic documents in some way or another, the argument can again be made that markup is essential to its concerns.

In this chapter I focus on markup in a yet more specific sense of the word, so-called generalized markup. Before proceeding to explain more precisely what is meant by the term in this more specific sense, however, it may be instructive for readers who are unacquainted with generalized markup to note that it is an essential part of a number of high-quality document processing and publication systems. It is also an important part of the technology underlying the World Wide Web (henceforth referred to as "the web").

Today the production of any scholarly edition inevitably relies on the use of computer-based publication methods, and thus also markup, whether those responsible for the edition are aware of it or not. The most popular text-processing systems in use today, off-the-shelf word processors and web or print publication systems, do not make use of generalized markup. Moreover, they take care of the markup "in the background" and allow the user to interact directly with documents displayed in the intended (or at least the resulting) visual layout (so-called WYSIWYG, or "what you see is what you get").

The kind of use of generalized markup discussed here, however, exposes the markup to the user and forces him at least sometimes to work with documents in an intermediate form that is visually much less close to the finished, printed, or displayed result. The path from the initial work at the keyboard to the finished product may seem—and indeed mostly is—more cumbersome and time consuming than it may be by using off-the-shelf software. If the sole aim of preparing an electronic document is the production of the visual result on screen or paper, adoption of generalized markup certainly does not save intellectual effort, work, time, or money.

However, if the aim of preparing the edition is extended to include making it available for use with modern tools for search and retrieval, computer-assisted concordancing, collation, word-frequency counts, collocation analysis, linguistic or stylistic analysis, and so forth, then in practice there is no alternative to employing standards for generalized markup. Thus, even in cases where the editors or the primary target audience themselves have no intentions to employ methods of search or analysis, the documents will be readily available for such use by the increasing number of readers who do have such intentions.

Perhaps equally or more important, documents prepared adhering to standards for generalized markup have better chances of long survival in a world of ever-changing computer systems, software, and formats. Many are those who have suffered the experience of discovering that documents that have been excellently prepared with easy-to-use off-the-shelf software become increasingly difficult to access—and finally, in practice, completely unusable—as the technology supporting the format they were prepared in becomes obsolete. The digital world, in which software and storage formats continuously change, is in many ways an ephemeral world. Preparing documents using standard generalized markup is the best way of ensuring usability and maximizing the chances that documents prepared today can still be read and used in a more distant future. Thus, one of the main reasons for recommending generalized markup is longevity.

On the other hand, the use of markup is always associated with underlying assumptions about the nature of texts and of textual editing. All too often such assumptions are made only silently, without explicit reflection, and uninformed by the specific nature of textual scholarship. It is only when these underlying assumptions about the nature of the object and task at hand are made explicit and the markup system adapted to the basic theoretical presuppositions and practical requirements of textual scholarship that markup can be put to its best use for the purposes of such scholarship.

Different editorial approaches vary not only in the methods of establishing the text but even in their basic assumptions about what they are trying to represent—that is, the object of their study. While *genetic editing* aims to record the sequence of scribal acts of which the manuscript is the tangible result, the *copy text* tradition aims at establishing a text that is as close to the author's intentions as possible. Whereas the *diplomatic edition* aims to accurately represent a manuscript down to grapheme and even allograph level, the *normalized edition* contents itself with a morphologically correct and "normalized" representation where spelling errors and slips of the pen have been silently corrected. While the *best text* approach usually contents itself with representing one witness and records variants in an apparatus, so-called *eclectic editions* establish a main text that is a selection from several or all of its witnesses.

There is no markup system that supports the requirements of all these editorial methods equally well. The strength of generalized markup is that the markup system can be adapted to suit the needs of each individual editorial effort in the best way possible. Of course that does not mean that generalized markup provides the answer to all editorial problems. But it does force each editorial endeavor to make its methods and its requirements explicit, and, to the extent that this is possible, to make them explicit in a way that lends itself to rigorous formal treatment. This incentive to explication of editorial methods may in itself be a good thing. Even so, large parts, and perhaps the

essential parts, of editorial scholarship do not lend themselves to rigorous formal treatment. What markup systems can help with is to make it easier for the textual scholar to concentrate on those parts rather than the more trivial aspects of the work.

All of this is to say that markup technology is a useful tool for textual scholarship. However, the opposite argument may also be made: textual scholars probably know more than anyone else about the richness of the various technologies that have supported transmission of texts down through history, in and between different literate cultures. Maybe this is why textual scholars and other representatives of the arts and humanities have already been able to contribute a great deal to markup theory and to the development of markup systems. So an additional reason for textual scholars to engage in work on markup is altruistic: through their participation, the shaping of future text technology may benefit from their knowledge rather than being left entirely to computer and information scientists, software developers, industry, and commerce.

Some Remarks on Terminology

Before we proceed further, three remarks on terminology are probably in order. First, as already mentioned, this chapter focuses on matters that are assumed to be of particular interest to textual scholars. The term "textual scholarship" is sometimes understood in a rather narrow sense: textual scholars are assumed primarily to be concerned with scholarly, critical, or documentary editing of monuments of the literary canon. Sometimes, however, textual scholarship is taken to also include various other sorts of scholarly or scientific study of almost any kind of text, ranging from ancient and modern literary studies to, for example, quantitative analyses of nonliterary texts. For the purpose of this chapter, the emphasis is on the former, more conservative understanding, although I have tried to also take into account the wider audience indicated by the latter understanding.

Second, the terms "text" and "document" are often used interchangeably and without difference in meaning in the literature on markup, and so are compounds like "document markup" and "text markup." For most text scholars, who are used to distinguishing carefully between a document and its text, or between a text and its states or witnesses, this fact alone may be enough to give rise to some skepticism. Although admittedly this terminological conflation is sometimes the result of unawareness of the importance of the distinction, most markup theorists are acutely aware of the problem.

But the distinction, which has traditionally been honored by textual scholars, has turned out to be difficult to adapt to electronic documents in general and to the use of generalized markup in particular. For example, an electronic docu-

ment is not a physical object on par with books, sheets of paper, or clay tablets, and it has no weight and no colors; rather, it consists of a certain constellation of electromagnetic or other kinds of traces or marks on some carrier. And when one talks about an electronic document, it is usually not even this physical substrate that is referred to, but rather the pattern itself on some more abstract level, such as a bit sequence, a character sequence, or the like. (Ironically, the term "text file" has become the standard term for computer files of a specific format.) This is not the place for an in-depth discussion of such issues, however. Instead, I have used the term "document" indiscriminately throughout the rest of this chapter, except where it has seemed important to signal that "text" in some more abstract sense is what is discussed.

Finally, what the English term "markup" refers to in this chapter is in other contexts sometimes called "encoding," "coding," "tagging," or "annotation" (corresponding terms in German are *Auszeichnung* or *Kodierung*; in French, *balisage*). The term "markup" itself was originally used to refer to marks or remarks made on the physical copy of a document to communicate instructions about modifications to be performed in the form of corrections, intended typography, and other details of concern to authors, editors, and printers, and thus carries associations to conventional editing and printing. The terms "encoding" or "coding," however, may carry associations to the practice frequent in social sciences and psychology of mapping instances of more finely differentiated data onto a limited set of values. The term "annotation" is typically used in linguistics, where linguistic data are frequently enriched with information about their linguistic properties, such as part of speech category, lexical forms, case, gender, and so on.

In the following section I implicitly try to introduce a more systematic terminology, but no attempt is made to relate this to the unfortunately rather confusing multitude of variant terminologies found in the literature on the subject.

Markup and Markup Standards

What Markup Is

A digital document may be seen as a linear sequence of discrete characters. Each character is associated with a letter or other sign from a given writing system, but each is also assigned a numeric value. Thus, characters (letters) and character strings (words) may easily be identified, counted, sorted, and manipulated by computer programs.

However, documents are not just linear sequences of characters. Scholarly editions, for example, generally have a complex, nonlinear structure, including cross-references, one or more apparatuses with variant readings, and so on; and

the manuscripts from which they are made may contain inter- and intralinear insertions, marginalia, deletions, substitutions, and so on. This may be one end of a spectrum ranging from very simple to extremely complex documents, but few documents are entirely linear. For example, modern business documents and technical manuals also contain nonlinear features such as tables, cross-references, footnotes, endnotes, and marginalia. In general, typography, letter size, color, and other kinds of visual layout are important aspects of all kinds of manuscripts as well as printed documents (and, indeed, also of electronic documents presented on the computer screen).

None of this easily lends itself to representation in the form of linear sequences of characters. For this reason it soon became usual practice to insert reserved character strings into electronic document files in order to represent features of the documents that could not readily be represented as letters, punctuation marks, and other visual features. In analogy with editors' and proofreaders' marks on manuscripts in preparation for traditional print, these reserved character strings came to be called markup.

For example, the sentence "J. S. Bach was a great composer" could be marked up like this:

```
.bold J. S. Bach
.endbold was a
.italics great
.enditalics composer.
```

Or like this:

```
$b\J. S. Bach\ was a $i\great\ composer.
```

Or in any other way that allows a computer program to assign to it the desired layout and typography.[3]

We may thus define markup as the use of reserved character strings inserted into the sequence of characters of electronic document files in order to denote or signal features of the document that cannot readily be conveyed by characters directly representing its verbal content.

In general, a *markup language* consists of an explicitly defined markup *syntax* and a controlled markup *vocabulary*. As such, a markup language is an essential part of any *markup system*, which in addition to the markup language also contains methods, tools, and associated practices for exploiting the markup.

The Need for Standards

Standards can sometimes and for some uses be helpful, but at other times and for other uses they may be harmful. Generalized markup is one of the areas where standardization at a certain basic level has proved beneficial for quite general

purposes, though when enforced at a somewhat higher level it is detrimental, at least to purposes of relevance for textual scholarship.

In the early days of computer-based document processing there was no generally accepted standard for markup. Most markup systems were directed toward capturing and controlling the visual appearance of documents rather than taking advantage of new opportunities provided by digital media. Documents with this kind of *procedural* or *presentational* markup were well suited for computer-assisted printing, but less well suited for retrieval, linguistic analysis, and other uses that are peculiar to digital documents.[4]

Software producers used their own mutually more or less incompatible markup systems in the form of proprietary file formats. The functionality of the markup was tightly dependent on the specific software, which kept the markup invisible to the user. For a long time the software houses seemed to regard their markup systems as strategic means for holding on to their customers. The multitude of incompatible systems and the lack of publicly available documentation made the exchange and reuse of electronic documents as well as document-processing software difficult, time-consuming, and expensive.

The result was considerable expense and inconvenience for users in general, but quite possibly it created an even greater problem for textual scholarship than it did elsewhere. In other contexts, documents may function primarily as a medium for the transmission of information about some subject matter that is external to the document. In textual scholarship, however, the object of study frequently is the document itself. In other contexts, documents may be of interest for only limited periods of time, yet textual scholars often work with documents containing texts that are transmitted over hundreds or even thousands of years. For humanities research in general and for textual scholarship in particular, it is therefore important not only to facilitate the exchange and reuse of electronic documents for the purposes of those documents' authors and intended users but also to ensure that the documents are preserved in a form that will make them accessible to present and future research and scholarship.

In addition, textual scholarship often relies on software that is specially developed for its own specific purposes, such as manuscript transcription and collation, critical editing, production of thesauri, and grammatical and lexical analyses. The difficulty and expense of developing such software was increased by the costs of maintaining it and ensuring that it could be used on documents stored in various and ever-changing proprietary formats. Textual scholars and the institutions responsible for preserving the sources and results of their work, such as archives and libraries, were among the first to encourage standardization of markup languages.

Therefore, considerable effort came to be (and still is) invested in the development of common standards for document markup. Not only public

institutions and some of the major players in the computer industry but also scholarly organizations from the humanities threw their support behind these developments, the principal aim of which can be described as improved efficiency in the production and distribution of electronic documents and the associated software.

Standard Generalized Markup Language (SGML)

What follows in this and the next section is a brief introduction to two markup systems, SGML and HTML, the first of which only a subset is still widely in use, and the second not at all recommended for use as a primary representational format in textual scholarship. They are introduced here, however, because they are both precursors of XML (which will be introduced later), and because I believe in the helpfulness of historical background to the proper understanding of the present, also in a context like this.

One important outcome of the standardization efforts described above was the adoption of SGML as an ISO (International Organization for Standardization) standard in 1986.[5] Strictly speaking, SGML is not itself a markup system or even a markup language, but a markup syntax specification—that is, a set of syntactic rules for how to construct markup languages.

Documents complying with markup languages that are constructed according to this syntax consist of various elements, each of which may be associated with document features such as layout (e.g., pages, columns, lines, font type and size), composition (e.g., chapters, sections, paragraphs; acts, scenes; cantos, stanzas, lines), linguistic structure (e.g., sentences, parts of speech), or thematic content-related information (e.g., names of persons, places, or organizations, bibliographical references).

Here is an example of how our simple sentence "J. S. Bach was a great composer" could be marked up in SGML:

```
<s><name>J. S. Bach</name> was a <emp>great</emp> composer.</s>
```

Whereas most earlier markup systems had been designed for presentational or procedural markup, SGML encouraged so-called *descriptive* or *declarative* markup. In this example, the markup indicates that the entire string is a sentence ("s"), that one part of the string is a name, and another part of it is emphasized ("emp"). In other words, the markup indicates what kinds of objects these objects are rather than indicating their intended or actual typographic features.

The SGML syntax as such, however, does not dictate which features of a document should be marked up. This is decided partly by the author or other person responsible for the actual markup of any specific document and partly by

the person who has designed the specific SGML markup language in question. In SGML terms this amounts to defining a document type definition (DTD). A document type definition specifies a markup vocabulary as well as rules for the ordering and containment relations between elements.

The introduction of SGML had a number of advantages compared to the earlier situation described above. One advantage was that any SGML-compliant document could be processed with software written for SGML. Since SGML was public and well documented, users were free to design their own markup languages and were not restricted to use software from any particular vendor. Another advantage was that not only did SGML provide a means for specifying quite precise rules for document structure adapted to the needs of individual users, but with SGML-aware software it was also possible to check automatically whether any given document complied with the rules specified.

However, the adoption of SGML progressed only slowly. The most important reason was likely its complexity. SGML included a number of optional features that turned out to be not widely used or requested. Partly because of this, and partly because of other design features, writing software for SGML was quite demanding.

HyperText Markup Language (HTML)

Since 1993 the propagation of SGML received a boost from the explosive growth of the web. The document standard used on the web, HTML (HyperText Markup Language), is based on SGML.[6] Therefore, it might be claimed that the popularity of the web also represents a success for SGML.

Even so, HTML had a number of drawbacks in the form of peculiar characteristics that conflict with many of the fundamental ideas underlying SGML. First, during the first ten years or so, web browsers were not designed for reading any form of SGML documents other than HTML. In effect, this meant that readers as well as authors were confined to this particular markup language and unable to make use of the flexibility to define markup for individual purposes, which had been central to the whole idea of SGML.

Second, the producers of web browsers started to compete by adding new "features" to HTML. In other words, HTML itself started to change, or to coexist in several different versions. The result was that even today web documents may be accompanied by remarks such as "Best read with [Browser X (version *n* or later)]."

Third, the opportunities for automatic checking were only rarely exploited. The result was that a large number of the documents on the web were not valid—they did not actually comply with the rules defined by HTML and SGML.

Fourth, HTML was entirely directed toward a (rather simplistic) visual presentation of documents and contained no or only very limited means for the representation of aspects such as their compositional, grammatical, dramatic, poetic, thematic, or editorial structure.

Later HTML has been modified to emphasize the representation of compositional structure of documents and extended with means for much more sophisticated layout and typography. Although HTML is still entirely unsatisfactory for the purposes of textual scholarship, it now offers a viable format for the visual presentation of such scholarly editions in electronic form.

These drawbacks motivated a search for alternative ways to transfer SGML documents via the web. It was against this background that work was begun on XML (Extensible Markup Language). The aim was to combine the simplicity of HTML with the expressive power and flexibility of SGML.

Extensible Markup Language (XML)

The World Wide Web Consortium (W3C) published XML as a W3C Recommendation in 1998.[7] Roughly, XML is a simplified subset of SGML. XML has gained considerable popularity, and it is probably fair to say that today it is favored by most practitioners of markup. This section gives a very brief introduction to the basics of XML.

THE BASIC ELEMENTS Let us once again take the simple sentence "J. S. Bach was a great composer" as our starting point. Here is an example of one way this sentence could be marked up in XML:

```
<s><name>J. S. Bach</name> was a <emp>great</emp> composer.</s>
```

In XML the angle bracket characters "<" and ">" are *delimiters* marking the beginning and end of a tag. A tag may be either a *start tag*, like "<s>", or an *end tag*, like "</s>". The strings inside the tags ("s", "name", and "emp") are called *generic identifiers*. The character string delimited by a start tag and the corresponding end tag is called an *element*, and the character string between these start and end tags is called *element content*.

Thus, the element "<name>J. S. Bach</name>" has the start tag "<name>", the element content "J. S. Bach", and the end tag "</name>". Because the generic identifier of the element is "name," the element is said to be of type *name*.

What XML defines is the delimiters and the way we are supposed to use them to compose tags and elements and so on. However, there is nothing in XML as such that dictates we must mark up the sentence in this particular way or that we must apply the generic identifiers used here. If we want to, we may just as well mark up the sentence as follows:

```
<line><bold>J. S. Bach</bold> was a <italics>great</italics>
  composer.</line>
```

In the first example, the identifiers are mnemonic shorthands for sentences, names, and emphasized strings, whereas in the second example, they stand for lines, bold print, and italic print. It is considered good practice to use generic identifiers that give some indication to the human reader or user of the document as to what they stand for. Anyone using the first kind of markup is probably marking up the semantic structure of the document, whereas someone using the second kind is probably more concerned with its visual appearance.[8]

XML offers a mechanism that may allow one to combine both views in a case like this. For example, we could choose to mark the sentence up as follows:

```
<s rend="line"><name rend="bold">J. S. Bach</name> was a <emp
  rend="italics">great</emp> composer.</s>
```

In this example, each start tag has been enriched with an *attribute* ("rend," for rendition) that has been given different values in each case ("line," "bold," and "italics").

Attributes are useful for many purposes. We might also want to add information to the "name" element about what kind of name it is (personal name, place name, etc.) and what the full normalized form of the name is. This may be achieved, for example, by adding attributes to the name element as follows:

```
<name rend="bold" type="person" full="Bach, Johan Sebastian">J. S.
  Bach</name>
```

In addition to elements, there is one more markup construct that any user of XML needs to know: *character entities*. As many users will have experienced, it is sometimes difficult to ensure correct transfer of non-Latin characters between different computer systems, or even between applications on the same computer.

The "special" Norwegian letters "æ," "ø" and "å," for example, often come out wrong. XML offers the possibility of representing them by character entities, such as "æ", "ø", and "å" instead. Thus, the name "Pål Færøy" may look like this in XML:

```
"P&aring;l F&aelig;r&oslash;y"
```

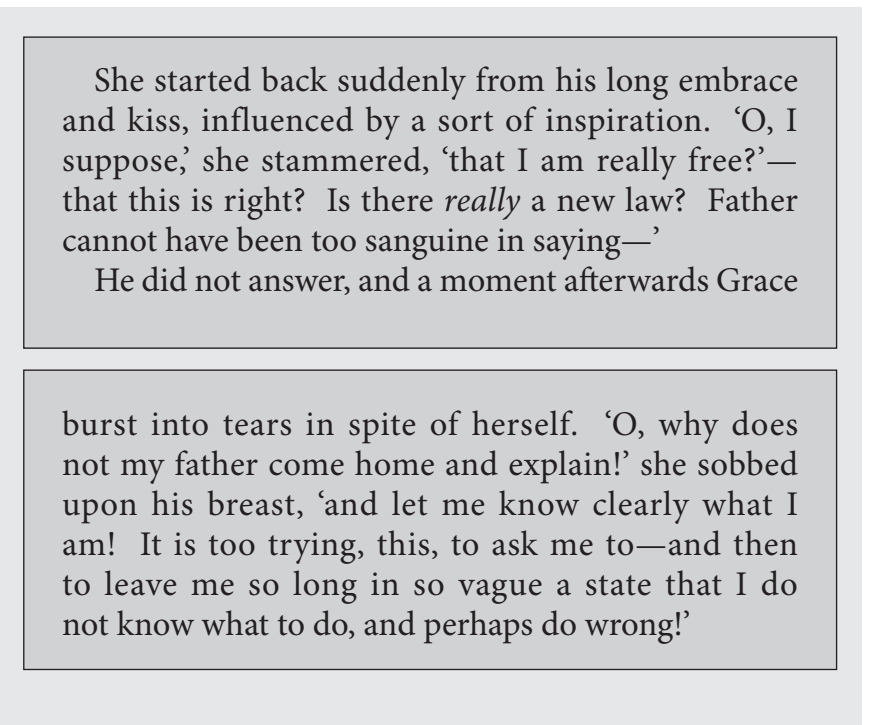
Admittedly, this encoding makes the XML form of the name considerably less readable to the human eye. But it has the important advantage that the letters can be transferred and displayed correctly by any XML-aware program. And, as is sometimes said, XML documents are not primarily meant for a human to read but for computer programs to process into readable form.[9]

XML entities can be used for other purposes as well, and XML contains mechanisms that have not been mentioned here. Even so, what has been outlined above should be enough for anyone to understand the basics of an XML document when they see one.

DOCUMENTS Consider the following facsimile of the bottom of a page and the top of the following page of a hypothetical reprint of the 1896 edition of Thomas Hardy's *The Woodlanders* (see fig. 5-1).[10]

What would a transcription of this fragment in XML look like? Again, XML as such does not dictate exactly how this should be done. However, here is one example of how we could go about it:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE doc SYSTEM "file:/D/Book/2009/Aurora/hardy01.dtd">
<doc>
<front>
          <author>Thomas Hardy</author>
          <title>The Woodlanders</title>
          <edition>Hardy,Thomas: The Woodlanders, Osgoood,
             McIlvaine, London 1896.</edition>
</front>
<body>
<!-- Beginning of excerpt from pages 365-366 ... -->
```


She started back suddenly from his long embrace and kiss, influenced by a sort of inspiration. 'O, I suppose,' she stammered, 'that I am really free?'—that this is right? Is there *really* a new law? Father cannot have been too sanguine in saying—'

He did not answer, and a moment afterwards Grace

burst into tears in spite of herself. 'O, why does not my father come home and explain!' she sobbed upon his breast, 'and let me know clearly what I am! It is too trying, this, to ask me to—and then to leave me so long in so vague a state that I do not know what to do, and perhaps do wrong!'

Figure 5-1: Fragment from the facsimile of the bottom of page and the top of page of a hypothetical reprint of the 1896 edition of Thomas Hardy's *The Woodlanders*.

```
<tab><para>She started back suddenly from his long embrace<lb/>and
  kiss, influenced by a sort of inspiration. <ds>O, I<lb/> suppose,
  </ds> she stammered, <ds>that I am really free?&mdash;<lb/> that
  this is right? Is there <emp>really</emp> a new law? Father<lb/>
  cannot have been too sanguine in saying&mdash;</ds><lb/></para>
<tab><para>He did not answer, and a moment afterwards Grace<lb/><pb
  no="366"/></para> burst into tears in spite of herself. <ds>O, why
  does</lb> not my father come home and explain!</ds> she sobbed
  </lb> upon his breast, <ds>and let me know clearly what I<lb/> am!
  It is too trying, this to ask me to&mdash;and then<lb/> to leave me
  so long in so vague a state that I do<lb/> not know what to do, and
  perhaps do wrong!</ds><lb/></para>
        <!-- ... end of excerpt. -->
</body>
</doc>
```

The first line is an XML declaration, indicating which version of XML and which character set has been used. The second line is a document type declaration, which will be explained in the next section.

The document proper (or, in XML terms, the *document instance*) starts with the start tag "<doc>" and ends with the end tag "</doc>"—meaning the document is an XML element of type `doc`. As in all XML documents, elements are linearly ordered and hierarchically nested inside each other. The doc element contains two further elements: `front` and `body`. The front element, in turn, consists of an `author`, a `title`, and an `edition` element, while `body` consists of two `para` (paragraph) elements. The `para` elements contain `ds` (direct speech), `emp` (emphasis), `lb` (line break), and `pb` (page break) elements.

The example contains two mechanisms not discussed earlier. First, the line starting with "<!--" and ending with "-->". These markers indicate that the enclosed character string is an XML comment. Comments are simply supposed to be disregarded by XML-compliant software. Second, instead of marking lines and pages of the original document as "line" and "page" elements, we have marked line endings by `lb` (line break) elements and the start of a new page by a `pb` (page break) element. Note that their delimiters "<" and "/>" indicate that the `lb` and `pb` tags are neither start tags nor end tags. They are empty elements, marking points rather than spans of the document.

Empty elements are often used in order to mark parts of the document that do not fit into the general hierarchical organization required by XML. (If we had decided to mark lines as elements with content in the example above, for example, we would have faced the problem that some of the `ds` (direct speech) elements start inside one line and end inside another, thus causing line elements and `ds` elements to overlap.) Empty elements are also used for recording phenomena such as images or hypertext anchors, which occur at specific points rather than spanning any part of a document.

The fact that XML documents must be element hierarchies—that is, that elements cannot overlap—has been the subject of much discussion. It does not mean that overlapping phenomena cannot be represented in XML. But the mechanisms for doing so are often cumbersome, and it is not easy to make XML software recognize such mechanisms.[11]

Well-Formedness

An XML document is organized in the form of a hierarchy of elements. The document element (in this case the `doc` element), which is commonly referred to as the *root element*, contains further elements, which in turn may contain yet further elements, and so on, until we finally reach a "bottom" level of elements containing no further elements, either in the form of empty elements or in the form of `PCDATA`—that is, character strings (often informally referred to as document content or even as "the text itself.")

Roughly, any document that adheres to these very basic syntactic rules is said to be *well formed*. It is a simple but also very important requirement that all XML documents should be well formed in this sense. If they are, any XML-compliant software will be able to parse the document—in other words, it can recognize its element structure and other mechanisms. If documents are not well formed, there is no guarantee that they behave as expected when processed by XML-based software.

Validity

For some purposes, tighter control over the structure of documents is required. This is the point at which we may return to the second line of our example from the previous section—its document type declaration—which reads:

```
<!DOCTYPE doc SYSTEM "file:/D:/Fag/2009/Aurora/hardy01.dtd">
```

This line declares that the root element of the document is of type `doc`, and it also informs us that a *document type definition* (DTD) for this element type can be found on the system resource referred to as "file:/D:/Fag/2009/Aurora/hardy01.dtd".

As mentioned earlier, a document type definition specifies a markup vocabulary—that is, a set of generic identifiers—and rules for the ordering and containment relations between elements of different types. In our example, the DTD might look like this:

```
<!ELEMENT doc(front,body)>
<!ELEMENT front(author,title,edition)>
<!ELEMENT author(#PCDATA)>
```

```
<!ELEMENT title(#PCDATA)>
<!ELEMENT edition(#PCDATA)>
<!ELEMENT body(para)+>
<!ELEMENT para(#PCDATA | ds | lb | pb | emp)*>
<!ELEMENT ds(#PCDATA | ds | lb | pb | emp)*>
<!ELEMENT emp(#PCDATA | ds | lb | pb)*>
<!ELEMENT lb EMPTY>
<!ELEMENT pb EMPTY>
<!ATTLIST pb no CDATA #IMPLIED>
<!ENTITY mdash '—'>
```

The lines starting with "`<!ELEMENT`" can be read as follows: "`<!`" signals the beginning of a declaration, and the key word "`ELEMENT`" tells us that indeed what is declared is an element. The key word is immediately followed by the generic identifier of the element, `doc`. The parentheses contain the element's *document model*. In this case, the document model—(front, body)—states that the `doc` element must contain one `front` element followed by one `body` element.

The content models of the various other element declarations tell us, for example, that `author` may contain `PCDATA` but no further elements, that `body` must contain one or more `para` elements, and so on. The last two lines declare the attribute `no` of the `pb` element and the character entity `mdash`.

As can be seen, the rules formulated in the DTD are considerably tighter and more precise than what is required for well-formedness alone. Any document conforming to the rules declared in this DTD is said to be valid according to the DTD.

It is sometimes said that DTDs are not required in XML. This is true in the sense that any well-formed XML document can be correctly parsed by XML software without access to its DTD. It is also true in the sense that alternative formalisms for document validation exist. The general term for DTDs and other formalisms for specifying constraints on XML documents is *schema languages*. Schema languages are extremely useful for a number of purposes, especially for document creation, editing, and quality assurance.

Data Structure

The elements directly contained by an element are called its *children*. Taken together, the children and any other elements contained by them are called the *descendants* of the element. Conversely, the element directly containing an element is called its *parent*. Taken together, the parent and any other element containing the parent are called the *ancestors* of the element. Two elements directly contained by the same element are called *siblings*.

In our example, the `doc` and `body` elements are ancestors of the first `para` element, while its `ds`, `emp`, `lb`, and `pb` elements are among its descendants. The

`body` element is its parent, while the next `para` element is its sibling. This terminology points us in the direction of another feature of XML documents that is considered extremely useful by many: the fact that any XML document can be represented as a specific kind of directed acyclic graph, a so-called *document tree*. The XML tree for our sample document is illustrated in figure 5-2.[12] The XML document tree illustrates a way of thinking about documents that is useful for at least two reasons: (1) because it is a visual representation of an abstract model of the various parts of a document and their relationships, a model that may be helpful in reasoning about document structures; and (2) because this kind of abstract model also underlies a data structure used for the internal representation of documents by XML-aware software applications.

In conversation with XML systems or application developers one will often hear references to document trees and their elements; the ancestors, descendants, siblings of these elements; and so on. Textual scholars who rely on XML applications or on developers of such applications are therefore well advised to acquire a basic grasp of this way of thinking about documents.

Transformations

At this point readers may understandably have become impatient. What is the use, after all, of representing documents in the form exemplified by the XML example above? It is verbose, it is hard to read, it is ugly, and it does not seem to provide any information that could not have been presented in a more readable form by conventional means of print typography and layout.

This is all true. The reasons for preferring an XML representation have been given above, and further reasons will be given later; in essence they have to do

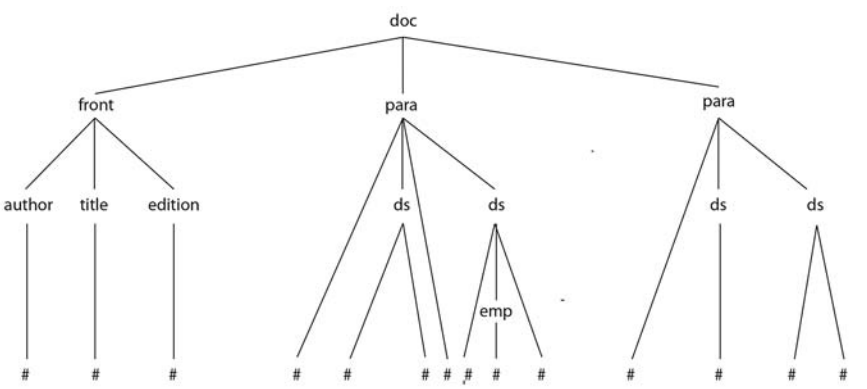


Figure 5-2: The XML tree of a sample document.

with increased usability, flexibility, and longevity. I have mentioned that XML documents lend themselves to automatic retrieval, concordancing, indexing, linguistic and stylistic analysis, and so on. None of this would be worth the effort, however, unless we could also produce conventional visual presentations of our documents, employing conventional means of visual presentation. Therefore, and because space does not allow for discussion of retrieval, analysis, and so forth, we now proceed to look at how XML documents can be transformed to presentational forms by the means of *style sheets*.

XML's style sheet language is called XSL (Extensible Stylesheet Language). XSL style sheets are typically designed for presentation of documents in different layouts and with varying degree of detail. The output can be formatted as PDF or PostScript for high-quality print or HTML for presentation on the web.

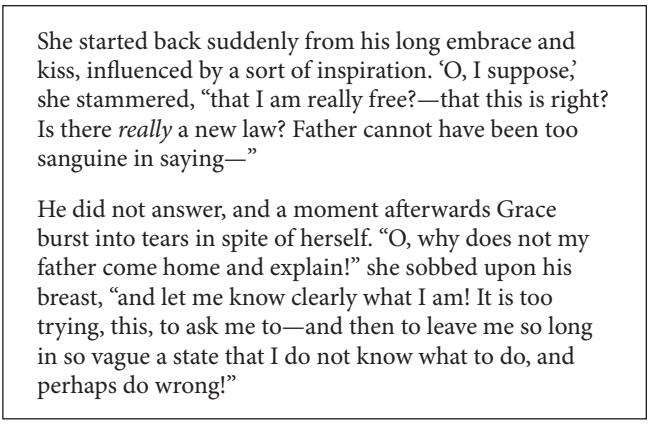
Here is an example of the most relevant parts of a very simple style sheet for our sample document fragment:

```
<xsl:template match="para">
          <xsl:element name="p">
             <xsl:apply-templates/>
          </xsl:element>
</xsl:template>
<xsl:template match="emp">
          <xsl:element name="i">
             <xsl:apply-templates/>
          </xsl:element>
</xsl:template>
<xsl:template match="ds">
          <xsl:text>"</xsl:text>
          <xsl:apply-templates/>
          <xsl:text>"</xsl:text>
</xsl:template>
```

It is not necessary here to go into any details of how style sheets work. Let it just be mentioned that of the three sections displayed above, the first one indicates roughly that `para` elements should be separated by blank lines, the second that `emp` elements should be printed or displayed in italics, and the third that `ds` elements should be printed with a double-quote open before and a double-quote close after. If we give our sample document and this style sheet to an XSL processor, it produces the output found in figure 5-3.

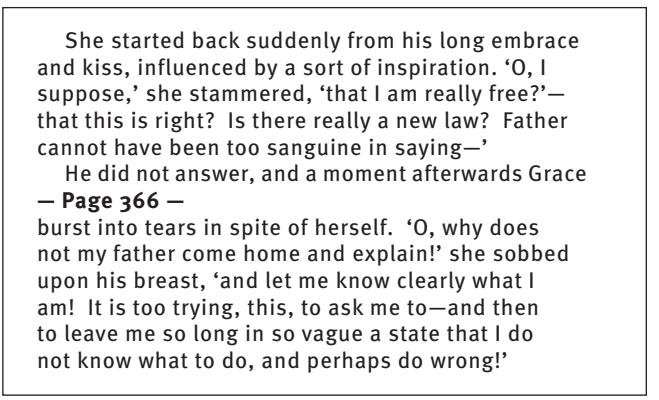
Alternatively, we might apply a different style sheet and have the sample printed in a different font, mark the original's page breaks, and retain its line breaks and its use of single quotation marks, as illustrated in figure 5-4.

This illustrates one of the advantages of preparing a document in XML: that it may subsequently be presented in a variety of different formats and styles without requiring interference with the XML source file in any way, thus both saving work and ensuring data integrity.


She started back suddenly from his long embrace and kiss, influenced by a sort of inspiration. 'O, I suppose,' she stammered, "that I am really free?—that this is right? Is there *really* a new law? Father cannot have been too sanguine in saying—"

He did not answer, and a moment afterwards Grace burst into tears in spite of herself. "O, why does not my father come home and explain!" she sobbed upon his breast, "and let me know clearly what I am! It is too trying, this, to ask me to—and then to leave me so long in so vague a state that I do not know what to do, and perhaps do wrong!"

Figure 5-3: Sample document using the XML code after being processed by the XSL processor using the XSL style sheet.


She started back suddenly from his long embrace and kiss, influenced by a sort of inspiration. 'O, I suppose,' she stammered, 'that I am really free?'—that this is right? Is there really a new law? Father cannot have been too sanguine in saying—'

He did not answer, and a moment afterwards Grace

— Page 366 —

burst into tears in spite of herself. 'O, why does not my father come home and explain!' she sobbed upon his breast, 'and let me know clearly what I am! It is too trying, this, to ask me to—and then to leave me so long in so vague a state that I do not know what to do, and perhaps do wrong!'

Figure 5-4: Sample document using the XML code after being processed by the XSL processor using another XSL style sheet.

The Text Encoding Initiative

XML provides a standardized syntax for how documents can be marked up, but it does not say a word about which features of a document should be marked. Supplying a semantics to the syntax—in other words, defining a vocabulary of tags used for marking elements and specifying what they mean and how they can be combined—is a task left to the users. While this certainly seems to allow for the kind of freedom and flexibility required in textual scholarship, where "[e]very textual situation is unique,"[13] it also seems to put a large burden of preparation on new users. And although nobody in their right mind would insist that every editorial project should be expected to use the same schema, record the same features, and adapt to the same basic document structure, there are enough similar phenomena to be recorded to make it seem a waste to have every project do roughly similar things in radically different ways.

One response to these dilemmas is provided by the Text Encoding Initiative. There certainly are editorial projects that choose to build their own encoding scheme from scratch, and there certainly is no lack of document schemas, but few of them answer the needs of textual scholarship. So even if the Text Encoding Initiative does not provide the answer to every question, it has gained enough recognition and is broadly enough used in order to deserve attention from anyone planning to embark on an editorial project.

Work on the Text Encoding Initiative began in 1987, just one year after SGML had been approved as an ISO standard. The *TEI Guidelines for Electronic Text Encoding and Interchange*, the result of a collaborative effort by researchers from a variety of humanities backgrounds, was first published in 1994 and has subsequently appeared in a number of revised and extended editions.[14]

The *TEI Guidelines* describe a markup system that consists of a series of DTD modules, or *tag sets*, which can be combined and adapted to special needs. A core tag set defines elements for features that are assumed to occur in practically any type of document. On top of the core tag set, users select one base tag set. There are four base tag sets to choose from: one for prose, verse, and drama; one for transcriptions of speech; one for print dictionaries; and one for terminological databases. In addition to the core and base tag sets, one can choose freely from a range of additional tag sets that serve various needs such as manuscript description, representation of primary sources, critical apparatus, linguistic analysis, and the like.

For many phenomena the TEI offers not just one but a variety of suggestions for how they may be represented. The *TEI Guidelines* discuss the pros and cons of these various approaches and encourage users to assess for themselves which alternative is best in each respective case.

The *TEI Guidelines* deal with almost every conceivable type of document, from papyri and palimpsests to modern office documentation and hypertext. Even so, it is stressed time and time again that the system should not be regarded as complete and that the need for further elaboration and modification should always be kept in mind. Indeed, the system includes provisions that allow it to be extended and modified by individual users.

What the *TEI Guidelines* offer is therefore an extensive yet liberal system that caters to a very broad range of document types and a large variety of textual phenomena. The system has a modular structure and permits modification. It has been widely applied in the humanities and is also used outside of academic circles. Since the conclusion of the developmental project in 1994, the number of people actively using the TEI system has grown into a considerable community. Their interests are now formally represented by the TEI Consortium, established in 2000, and further development of the system is an ongoing concern.

The fundamental features of the TEI's system—modularity, modifiability, numerous alternative means of handling analogous phenomena—can make it seem too complex in cases where the requirements are simple and the difficulties few. Therefore, the TEI has also developed a "light" version of its system, called TEILite.

Pros, Cons, and Perspectives

XML has become pervasive in nearly all kinds of document processing, ranging from publication on the web to high-quality print production. It has also found applications in a number of other areas of potential interest to textual scholarship, such as representation and processing of musical, mathematical, and chemical notation as well as graphics and multimedia documents. The many available tools for retrieval, collation, annotation, and analysis are also of obvious relevance for textual scholars.

The number and diversity of available systems, tools, and applications for XML-based markup is of course one of its strengths. At the same time, it may be a source of confusion and bewilderment among prospective users. The jungle of technicalities, standards, software packages, and acronyms may seem forbidding to someone new to the technology. It is a comforting fact, however, that the average user of XML does not need to enter this jungle.

Even so, if the immediate or even the ultimate aim of a project is a high-quality print edition, why not prepare documents with WYSIWYG editors for page-description languages like PDF or some other widespread proprietary format? After all, these tools are simple to use and at least as ubiquitous as XML. At least the following three reasons may be given: First of all, WYSIWYG tools are indeed available also for XML editing. The time is past when XML users had to work directly with "angle bracket notation." Second, converting a document from a proprietary format to XML may be difficult, unreliable, and sometimes impossible in practice, while conversion from XML to other formats is comparatively easy, as it is part of the whole idea behind XML. Third, proprietary formats are generally not publicly documented, because they bind users to commercial or proprietary software, and there is considerable risk that documents prepared in such formats will sooner or later become obsolete.

One complaint often made about XML is that there are certain document structures that are difficult to represent in XML. And this is indeed true: XML's insistence on hierarchical nesting of document elements does make it difficult to represent overlapping hierarchies, discontinuous or interrupted elements, and alternate orderings of document elements. On the other hand, there are well-known techniques for handling these problems in XML. The *TEI Guidelines*, for example, provide a number of such methods.

None of the known alternatives to XML combine a viable solution to these problems with the strengths of XML, such as the tight integration of notation, constraint language, and data structure; a host of general-purpose as well as specialized application software; and, most important, none of them are international standards that may be assumed to exist and be supported in the future.

There is no reason to believe that XML is the ultimate answer to all problems of document representation and processing or that it will last forever. There are good reasons to believe, however, that XML will stay with us for a long while. With the large investments already made in XML technology, there are also good reasons to believe that whatever the next generation of document technology might look like, it will have little chance of acceptance unless easy and lossless conversion from XML is made possible. Thus, adoption of XML in textual scholarship may also be in the interest of preservation.

Notes

1. I wish to thank Daniel Apollon, Alois Pichler, Tone Merete Bruvik, and Odd Einar Haugen for their useful comments and advice on earlier drafts of this chapter, the shortcomings of which they are of course in no way responsible.

2. This view is expressed in one of the earliest seminal papers on markup theory; see Coombs et al. 1987.

3. The examples are fictional. The first example uses full stops at the start of lines to signal markup and thus exploits the fact that full stops rarely occur at the beginning of a line in normal documents. The second example uses reserved characters "$" and "\" to separate the markup from the rest of the document. The observant reader will already have noticed that the markup in the examples does not allow a program to distinguish personal names from other items that may have been marked as "bold," or to distinguish emphasized words from other items that may have been marked as "italics."

4. This use of the term "presentational" is not strictly in accordance with the terminology as originally introduced in Coombs et al., where the visual layout itself is what is considered "presentational markup." It has become customary, however, to use "presentational markup" to refer to markup that records visual layout.

5. SGML: Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML), ISO 88791986, International Organization for Standardization, Geneva 1986.

6. This is a slight simplification of historical facts. HTML was not strictly defined as an SGML standard until sometime after its initial use on the web.

7. The World Wide Web Consortium's website can be found at http://www.w3.org/XML.

8. It is true that most practitioners of XML will recommend the first kind of markup rather than the second, and for good reasons. However, this is a fact about common usage of XML: there is nothing in XML as such that prevents the second kind of markup.

9. For a long time the number of characters on any given computer system was severely limited, and hardware and software suppliers used different conventions for the representation of identical character sets. The computer world is currently moving toward generally accepted standards that include the majority of characters of major past and present writing systems. Problems remain in this field concerning some writing systems that are clearly of interest to textual scholarship, but the issue will not be further pursued here.

10. We have made this facsimile for demonstration purposes.

11. See, for example, DeRose 2004 and Witt et al. 2005.

12. For the sake of simplicity, empty elements have been left out from this illustration.

13. Gaskell 1978, 6.

14. Sperberg-McQueen and Burnard 2002.