Some Problems of TEI Markup and Early Printed Books

Carole MAH, Julia FLANDERS and John LAVAGNINO

Résumé. Cet article traite de deux groupes de problèmes d'enregistrement de texte électronique, rencontrés par le *Women Writers Project* (WWP) de la Brown University. Le WWP construit une base de données reprenant le texte intégral d'œuvres imprimées avant 1830, écrites en anglais par des femmes écrivains. Nous utilisons le *Standard Generalized Markup Language* (SGML), suivant les *Guidelines for Electronic Text Encoding and Interchange* (Directives pour l'Enregistrement et l'Échange Électronique de Texte) publiés par le *Text Encoding Initiative* (TEI; Initiative d'Enregistrement de Texte). SGML est un système puissant pour représenter des structures complexes du texte. Cependant, cette représentation nécessite un enregistrement assez complexe. De plus, il faut réfléchir attentivement au but auquel les données vont servir. Nous présentons ici quelques méthodes pour traiter ces problèmes et une analyse des difficultés qu'elles suscitent.

Keywords: Text encoding, transcription, abbreviation, emendation, SGML, textual structure, TEI.

Mots-clés : Enregistrement de texte, transcription, abréviation, correction, SGML, structure textuelle, TEI.

Introduction

During the last ten years, the usefulness of SGML (Standard Generalized Markup Language (Goldfarb, 1990) in the transcription and encoding of a fulltext database has been put to practical test with the growth of an ever larger number of text encoding projects. Using SGML to build an on-line textbase means identifying and marking by name and related attributes all the salient structural elements in a given document. By doing so, the full power of the electronic medium can be utilized. Just a few of the benefits

✉ Brown University Women Writers Project; Box 1841, Brown University, Providence, Rhode Island 02912 (U.S.A.).
E-mail: Carole_Mah@brown.edu, Julia_Flanders@brown.edu, John_Lavagnino@brown.edu

include the ability to produce printed books, online access, CD-ROMs, on-demand printouts, and other products from a single source document; the ability to perform the most powerful and context-sensitive electronic searching; and the ability to provide widespread access to documents otherwise unavailable. Since the inception of the Text Encoding Initiative in 1987 and the subsequent publication of its SGML-conformant Guidelines for Electronic Text Encoding and Interchange, many humanities text encoding projects[1] have built TEI-conformant fulltext databases that display to varying degrees the practical truth of many of the long-asserted advantages of using SGML to encode literary and linguistic texts.

The Women Writers Project is one of the most ambitious of these projects. Unlike many, our scope is broad in terms of both genre and chronology. Our texts sample the whole range of writings by women (both British and colonial) in the English language before 1830. Most text encoding projects cover a single era (e.g., the Victorian Women Writers Project)—often a single writer (e.g., the Nathanael Greene Papers). In addition, while there are other projects as broad in scope (e.g., the Perseus Project), few of these also aim to do diplomatic transcription of specific source texts; some even have the express aim of providing fully edited documents. All these factors have meant some delay in the full realization of various derivative textbase products from the WWP. However, they have also given us a unique perspective. In a sense, the challenges we face are a microcosm of those faced by the humanities computing world as a whole as it grows towards the construction of digital libraries. Some of the most promising features and benefits of using SGML for text encoding also pose some of the most difficult challenges. By providing a thorough exploration of a few of the conundrums we have faced, the WWP can benefit not only other projects which may face the same particular problems, but also the field as a whole, by provoking thought about the nature of text encoding and the kinds of issues it raises. In what follows, we will first present an analysis of some transcription challenges which arise from the need to emend errors and expand abbreviations; then we will discuss the problems of dealing with multiple hierarchical structures in a single document. These two issues are representative of the kinds of challenges posed by the form of transcription we have undertaken: some having more to do with the kinds of information we need to encode, and some with the logic underlying SGML itself.

[1] For a list of several such projects, see Ide, 1991.

1. Transcription challenges inherent in the document: The problem of dual emendation and correction

1.1. Advantages of an SGML solution

In constructing our textbase, the WWP envisions that it will be used in the following ways: to get a diplomatic transcription of the source for scholarly work, to get a clear transcription for reading, to verify the accuracy of the transcription, and to do useful searching. These goals seem to fulfill many of the most common features people expect of an electronic textbase. In addition, they fulfill many of the features one expects from various kinds of printed scholarly editions. SGML is instrumental in making it possible to achieve all these goals, but also brings to light some challenges which deserve careful thought. These challenges are not introduced by SGML but are inherent to doing transcription of early modern printed books; using SGML merely foregrounds the issues.

In constructing our textbase, the WWP's policy is one of "diplomatic transcription". "Diplomatic transcription" involves transcribing the text of the document without making any emendations or corrections of apparent errors in the source text based either on one's own judgment or on other versions of the text. By contrast, it is standard practice when producing reprints or critical editions to correct apparent errors; reprints typically do not even consider it necessary to inform the reader which words were corrected. When producing a critical edition, a scholar produces corrections to apparent errors only after comparing the readings of a given word in a wide range of relevant documents in order to determine, based on this body of evidence, whether the apparent error is in fact an error (such as a printer's error or a singular misspelling on the part of the author) or whether it is a period spelling or an idiosyncratic spelling (but consistent across editions or works) specific to that author. One of the reasons a diplomatic transcription does not presume to correct apparent errors is that it is by definition a transcription of only a single version of a text. Another reason, when working with pre-Victorian literature by women, is that spelling was not standardized before the 19th century, and for women even less so given their lesser access to formal education.

A clear transcription for reading means a transcription that does not require the reader to wade through the difficulties of apparent errors and old printing conventions. This is an ideal transcription for the casual reader and for many teaching situations. In such a transcription, not correcting

apparent errors could lead to great confusion in understanding meaning in context in some cases. In addition, the audience for such a transcription might not be interested in the uncorrected version. The reader of such a transcription would also not be interested in or necessarily be able to understand conventions such as the printing (prior to about 1650) of certain letter combinations in an abbreviated fashion. For example, a character with a short horizontal bar, small letter, or other mark directly over it (not above and to the side as in a superscript) is called a brevigraph. A "y" with a mark of some sort over it can stand for "the", "thou", or "that." An "e" with a short bar over it can stand for the letter combination "en" or "em". Reproducing these brevigraphs as closely as possible for such an audience would be counter-productive.

SGML gives one the ability to simultaneously record both the apparent error and one's correction, without compromising the policy of diplomatic transcription, since (using the TEI <sic> element) the document's content remains unchanged, with one's correction recorded on the attribute value:

```
<sic corr="whether">mhether</sic>
```

In this way, both a reading version and a diplomatic version can be produced from the single source transcription. The programmer can produce the diplomatic version of a text by specifying that the content of all <sic> elements should be honored (producing in this instance "mhether") and can produce the reading version by specifying that the content of the <sic> elements should be ignored in favor of the value of the corr attribute (producing in this instance "whether"). The same argument goes for the transcription of abbreviations such as brevigraphs; both forms can be encoded simultaneously using the TEI <abbr> element with the expansion recorded on the attribute value:

```
<abbr expan="condition">códitió</abbr>
```

The diplomatic version of the text would then honor the content of the <abbr> element, giving the abbreviated form with the brevigraphs. The reading version would ignore the content of the <abbr> element in favor of the value of the expan attribute, giving the expanded form (*i.e.*, "condition").[2]

---

[2] We should add that in all these examples, to help make them clearer, we have displayed the brevigraph (ó or ē) as it appears in the source, rather than using an SGML entity reference (`&oacute;` or `&emacr;`), which is our actual encoding practice; as we attempt to show in this essay, the real intellectual problem is in the handling of abbreviation and correction, and it is distinct from the practical problem of representing individual characters.

Having used SGML for transcription, one can simultaneously or separately produce two versions—a diplomatic version and a reading version—from the single source document and can thus cater to a variety of audiences without duplicating one's labor. Traditional scholarly editing projects usually provide no such access, choosing one approach or the other, but making no attempt to provide both. For instance, Malone Society type facsimile reprints provide the exact facsimile of the original characters and do not provide expansions. In contrast many critical editions expand and modernize everything. Using SGML to create an electronic edition gives one the option to enlarge the number of audiences for one's texts. An electronic edition has at least the potential to serve a wide variety of audiences from the specialist in early modern printed books, to the linguist specializing in the Renaissance, to the well-educated generalist, to the undergraduate student, to the general public. (In fact, the attempt to cater to as many audiences as possible is another way, in addition to broad range in time and chronology, that the WWP distinguishes itself from other humanities text encoding projects. And again, this is both an asset and a liability.)

Using SGML also gives the reader a way to verify the accuracy of the transcription. Having both the apparent error and the correction at hand (for instance in an online version that incorporates both the reading and diplomatic versions) lets the reader knows that the error is inherent and was not introduced by the transcriber, especially since, in a textbase of this size, there will inevitably be some errors introduced by the transcriber and the reader would want to distinguish between the two. This is of course a problem for traditional scholarly work as well. For instance, a Malone Society reprint presents the doubtful reading as is in the body of the text, and in the front matter provides a table of the doubtful readings and the editor's corrections together with a reference to the page on which the doubtful reading occurs. This is so obvious a problem that it seems unnecessary to mention it except that people often assume or expect otherwise in an electronic environment.[3]

Apart from the original unexpanded forms' intrinsic interest for many scholars, having both the abbreviated form as well as the expanded form

[3] Users either assume the cynical view that computers are nothing but problems and will assist one in introducing even more errors than normal, or they will take the naive "computers are magic" view and somehow therefore believe that there will be no errors. Although undocumented and somewhat impressionistic, this is not a phenomenon to be overlooked.

of a word allows the reader, as with apparent errors, the ability to verify the transcription. One could perhaps dispense with transcribing the original abbreviated form and simply encode only the expansion; e.g., simply type "condition" rather than:

```
<abbr expan="condition">códitió</abbr>
```

Then one could provide (instead of textual markup) a scanned image of the original page for comparison. However, should people doubt the accuracy of our expansion of several "e"s with brevigraphs as "em" rather than "en" or vice versa, they might want to do a search for all abbreviated forms (`&emacr;`) in context, and make their own judgment. This can easily be done if both versions are encoded in tandem; comparing each of our expansions to a scanned image would be quite a laborious back-and-forth task.[4]

Verification is not the only reason to provide both the abbreviated and expanded forms of a given word and both the corrected and uncorrected forms of a given word. Powerful searching capabilities are the most oft-heralded feature of electronic documents and especially SGML-encoded ones (which provide for sophisticated context-sensitive searching based on the structural hierarchy of the document elements). Expansion provides the full form of a word, which is what most people expect to encounter when writing or using programs for useful searching of a text or an entire textbase. For example the Oxford English Dictionary is using the WWP textbase as a new corpus upon which they can do searches for occurrences of words that may supplant the current earliest attested usage of that word. This is a significant and powerful resource for them, since prior to the existence of such electronic textbases, they had to do their research by hand. Similarly, correction of apparent errors provides meaningful data for searching programs (e.g., a search across one or several textbases for the word "pickle" would not turn up the occurrence misspelled "qickle" unless that word were tagged appropriately).

---

[4] Another important point arises from the fact that most brevigraphs stand for different letter combinations depending on context. If they did not, one could just transcribe all letters with brevigraphs over them without attendant markup, leaving it to an automatic processing program to add the markup. However, it takes a human to decide whether a given abbreviation stands for one thing or another; for instance, as previously mentioned, an "e" with a brevigraph over it could stand for "em" or "en".

1.2. Simple Cases

For both corrections and expansions, then, the SGML approach is clearly a more robust one than the traditional non-electronic (and many non-SGML electronic approaches), allowing one to provide a variety of audiences with, in each case, two different readings of the same text. In addition, however, in many texts there are large sets of non-overlapping examples of both corrections and expansions; in these cases, providing all four possible permutations of the two is a simple matter in some. For example, consider: `exqectatió` where the "q" is a typesetter's mistake. There are four possible readings of this word:

`exqectatió`	(uncorrected, abbreviated)
`exqectation`	(uncorrected, expanded)
`expectatió`	(corrected, abbreviated)
`expectation`	(corrected, expanded)

To encode this, one could do the following:

```
ex<sic corr="p">q</sic>ectati<abbr expan="on">ó</abbr>
```

—a very straightforward application of the TEI tagset. From this single source document one could then derive all four possible readings of every such instance in the document using simple, unambiguous processing programs to produce four full different text versions.

One could in fact chose any number of such approaches depending on one's analysis of the likely audiences, the sophistication of available software, and the relative amount of additional labor involved in each approach. In contrast, in most traditional publishing situations, no one would even attempt to solve this problem in a way that would facilitate providing four separate full texts; rather, the inevitable result would involve being forced to chose which bits of information to lose—which would be least useful to a given (usually single) target audience. Therefore, in the traditional type facsimile situation the doubtful reading would be printed (perhaps with a table of doubtful readings in the appendix), and the abbreviations would remain unexpanded. this provides a basic diplomatic version. In fact this same choice might be made by many a TEI-conformant project. The point is that SGML gives one the option not to have this be the *only* choice, so that if chosen, it is not by default but by an analysis of labor, audience, and software.

1.3. Additional layers of complexity: an in-depth example

What does one do when the set of doubtful readings and the set of abbreviations overlap? That is (for example) what, if the character or characters involved in a doubtful reading are also characters that have brevigraphs? Given the complexity of the textual issues, in both the traditional and the SGML publishing situation, some information will be lost no matter what one does. The traditional solution would not differ in the case of such overlap as compared to cases in which there is no overlap. However, the variety of ways of nesting structural hierarchies using SGML means that there are many ways of solving such problems. Many of these solutions are excellent and the choice will depend on the intended audience(s). A few of the potential solutions should be avoided because the way in which they nest elements introduces unnecessary difficulties and because they leave too much up to the processor rather than making as much as possible clear in the encoding. These are issues a traditional publisher has neither the privilege nor the burden of facing.

A typical Women Writers Project example of such a complex situation[5] is shown below, taken from Foxe's Actes and Monuments (which contain a version of Anne Askew's Examinations [Askew, 1563]):

tēpēted

Fully expanded and corrected, this would be:

tempted

The four major readings of this word are:

---

[5] One could encounter even more complex situations than the one described here. Consider again the case of the typical critical edition. Not only does such a project often chose to expand all abbreviations (e.g., printing "the" instead of printing a "y" with a brevigraph over it), it may also choose to normalize all archaic spellings (e.g., printing "Jesus" instead of "Iesus", "private" instead of "priuate", "always" instead of "alwaies", etc.). Should one wish to normalize in addition to expanding and correcting, one would use `<orig>` and its `reg` attribute:

```
<orig reg="J">I</orig>esus
```

It is clear that a combination of all three tags (`<abbr>`, `<sic>`, and `<orig>`) could be even more challenging. However, a full explication of such a triple-layered example would be quite complex and lengthy, and a double-layered one, involving only `<sic>` and `<abbr>`, serves just as well to illustrate the challenges in doing this sort of encoding. The remainder of this discussion focuses on such an example.

tēpēted	(uncorrected, abbreviated)
tempemted	(uncorrected, expanded)
tēpted	(corrected, abbreviated)
tempted	(corrected, expanded)

If the intended audience made it necessary to provide only one or two readings, it would of course be a trivial matter to encode this word with a simple encoding[6] such as either

```
t<abbr expan="em">ē</abbr>p<abbr expan="em">ē</abbr>ted
```

or

```
<abbr expan="tempemted">tēpēted</abbr>
```

in order to get either of the following two readings

tēpēted	(uncorrected, abbreviated)
tempemted	(uncorrected, expanded)

To get the first one (no expansion) the processor would simply ignore the attribute values on <abbr> whereas to get the second one the processor would heed it. Similarly, the simple encodings

```
tēp<sic corr="">ē</sic>ted
```

or

```
<sic corr="tēpted">tēpēted</sic>
```

would yield either of the following two readings:

tēpted	(corrected, abbreviated)
tempted	(corrected, expanded)

by taking or not taking the value of the corr attribute on the <sic> element.

A more complex possible encoding is (let us call it Example 5):

```
t<abbr expan="em">ē</abbr>p<sic corr=""> ✂
                              <abbr expan="em">ē</abbr></sic>ted
```

One could produce the following readings from this encoding:

tēpēted	(uncorrected, abbreviated)	[using **content** of <sic>; **content** of <abbr>]
tempemted	(uncorrected, expanded)	[using **content** of <sic>; **attribute** of <abbr>]
tēpted	(corrected, abbreviated)	[using **attribute** of <sic>; **content** of <abbr>]
tempted	(corrected, expanded)	[using **attribute** of <sic>; **attribute** of <abbr>]

[6] Which encoding one chooses in this case makes little difference.

How do these readings derive from the encoding? Two assumptions are at work here; the first is one made by the programmer; the second is an unavoidable result of the nature of an SGML-encoded document's binary tree structure. First, assume that each reading is produced by treating all instances of a giving element in the same way. That is, if the decision is made to take the attribute value on an element and therefore ignore its content, then all instances of the element are so treated by the programmer (this is a natural and easy assumption to implement since it requires no special actions on the part of the programmer). Second, assume that when tags nest, the outer tag (or "parent" element) takes precedence over the inner (or "child" element). Thus, in the second nesting, no matter what we had as the content of the second <abbr> would be ignored in favor of the value of the expan attribute on it:

```
t<abbr expan="em">ē</abbr>p ✂
                          <abbr expan="em">NONSENSE</abbr>ted
```

This would also produce:

tempemted (corrected, expanded) [using **attribute** of <sic>; **attribute** of <abbr>]

Following this same logic,

```
t<abbr expan="em">ē</abbr>p ✂
       <abbr expan="NONSENSE"><sic corr="">ē</sic></abbr>ted
```

would produce:

tempNONSENSEted (corrected, expanded)
[using **attribute** of <sic>; **attribute** of <abbr>]

Now study another, deceptively similar-looking encoding (let us call it Example 6):

```
t<abbr expan="em">ē</abbr>p ✂
             <abbr expan="em"><sic corr="">ē</sic></abbr>ted
```

The only difference between this encoding and the previous one (*i.e.*, Example 5) is that the nesting of the second <abbr> and the <sic> is reversed. One might think this would make no difference. However, observe that this encoding produces the following readings:

`tēpēted`	(uncorrected, abbreviated)	[using **content** of `<sic>`; **content** of `<abbr>`]
`tempemted`	(uncorrected, expanded)	[using **content** of `<sic>`; **attribute** of `<abbr>`]
`tēpted`	(corrected, abbreviated)	[using **attribute** of `<sic>`; **content** of `<abbr>`]
`tempemted`	(corrected, expanded)	[using **attribute** of `<sic>`; **attribute** of `<abbr>`]

Closely examining the last of these reveals that:

`tempted` (corrected, expanded)

cannot be produced from this encoding, whereas it can be from the opposite nesting. As explained above, this is because the parent element takes precedence over the child. Now it should be clear that the only way to get "tempted" from this encoding is by not making the first assumption described above that all instances of a given element should be treated identically. Rather, the programmer would have to specify that the second `<abbr>` should be treated differently than the first:

`tempted` (corrected, expanded)
[using **attribute** of `<sic>`; **attribute** of first `<abbr>`, **content** of second `<abbr>`]

Here, in the first instance of `<abbr>`, the attribute value is taken, whereas in the second instance the content of the `<abbr>` element is taken.

The TEI-conformant textbase project might very well turn to the use of feature structures at this point (discussing this option is outside the scope of this article[7]—for WWP purposes the added functionality feature structures would provide do not seem to outweigh the complexities they would introduce); however, that choice aside, these examples exhibit several important points about encoding early printed books. One is the utter necessity, when choosing which encoding scheme to implement, that the choice be well-documented and consistently implemented. Implementing several of the possible choices in a single document would be unwise in the extreme. It is also clear that it is important to chose a scheme which depends as little as possible on how the programmer treats the encoded document in using it to produce various versions of the text and depends as much as possible on the clarity and simplicity of the information provided by the markup. That is, markup situations which might force one to treat different instances of an element in different ways should be avoided at all costs.

[7] For the authoritative discussion, see chapter 16 in Sperberg-McQueen and Burnard, 1994.

Finally, in coming to a decision about which encoding scheme best achieves the goals of multiple versions of a text from a single source document, the facilitation of searching, and the ease of verification of transcription, it is important to consider which produceable versions are the most desirable to likely audiences.

The current preferred solution at the WWP takes all of these points into consideration. In particular, we have concluded that the "temempted" reading is less desirable among our potential audiences than the other three readings, since it is likely that any reader who would want abbreviations expanded would also like the errors to be corrected. Further, we wanted to avoid the potential confusion of the sort of nesting explored in the above examples. Therefore, we settled on the following:

```
<abbr expan="tempted">tēp<sic corr="">ē</sic>ted</abbr>
```

This would make the following three readings possible:

tēpēted	(uncorrected, abbreviated)
tēpted	(corrected, abbreviated)
tempted	(corrected, expanded)

With this encoding, no special treatment of different instances of an element is necessary to produce the desired results. Finally and perhaps most significantly (both from the point of view of readers who might want to look at the raw SGML and of student transcribers learning how to encode), this solution is much less long-winded and much easier to understand than the ones above.

It is important to re-emphasize that the problem of concurrent provision of correction and expansion is an intellectual and scholarly challenge that exists independently of SGML's existence. Using SGML merely makes it possible (in the simple, non-overlapping case) to address the question rather than to avoid it, duck it, or be forced to ignore it for obvious want of a solution. In the more complex case SGML can introduce some difficulties but only because it also introduces the opportunity to provide multiple versions of a text from a single transcription—something that would not otherwise be possible. In the section that follows, we will examine a case where it seems to be the nature of SGML itself that creates the encoding challenge.

2. Transcription Challenges arising from the Use of SGML: The Problem of Multiple Hierarchies

2.1. Groundwork and Assumptions

The central, enabling assumption underlying the design of SGML as a system for text encoding is that all texts consist of ordered hierarchies, wherein all elements are fully nested within other elements, with no overlap. This assumption is built into SGML at the most basic level, and it makes possible the unique functionality of SGML: the ability to specify and regulate a document's structure, and then use that structure for complex activities such as searching, processing, document comparison, and the like. At the same time, it has been apparent at least since SGML began to be used to transcribe existing documents that a given text can—and frequently does—contain multiple hierarchies whose elements overlap one another, an obvious case being physical structures like pages and textual structures like paragraphs, which are structured independently of one another and frequently overlap. Such situations may derive from the very nature of the traditional printed book (as in the case just given), where a physical object with a certain architecture contains an abstract structure of an entirely different sort: a sequence of words which make up a text. In addition, though, such overlap may occur within the text itself, between such different structures as the grammatical syntax of the text and its poetic form. Although the textual structure of a document has an obvious usefulness for research and navigation, the physical structure is also of immense importance, particularly for researchers who study the relationship between the text and its physical embodiment.[8]

In a traditional printed book, these different structures are unobtrusively accommodated, in a manner which has arisen over the long history of book production. However, the activity of text encoding, by making explicit the relationship which governs a set of individual details, creates linkages which—within the SGML paradigm of nested hierarchy—get in each other's way. As Renear and others have argued,[9] these different structures derive from methodological perspective rather than from genre, a point which has

[8] See for instance McKenzie, 1981. Note that we are not talking here primarily of work on the appearance of the page, for which a facsimile image would be indispensable; we are thinking more of research which would examine the structural relationship between the text and its embodiment, and would rely on the encoding to make that relationship tractable to formal study.

[9] Renear, Mylonas, and Durand, forthcoming.

several consequences. First, it guarantees that overlapping structures will be evident in any document which is encoded to accommodate more than one methodology. And second, in order to assign precedence among the structures—to decide which ones to encode straightforwardly and which to encode using an alternate system—the textbase designer must decide which methodologies best represent the intended function of the textbase. An electronic text project which was content to limit itself to encoding a single hierarchy would be able to provide only the most restricted version of the text. For a project like the WWP, which is engaged in transcribing rare books for a scholarly as well as a more general audience, multiple hierarchies result unavoidably from the textual and physical features of the document required by this audience. This is not all, however; as Renear et al. point out, some methodological perspectives themselves require attention to multiple hierarchies, so that even limiting one's encoding to a single approach cannot always avoid engaging with problems of overlap.

2.2. Practical Approaches

The difficulties of accommodating multiple hierarchies have been apparent since the initial discussions of encoding guidelines for humanities texts which preceded the advent of the Text Encoding Initiative, and the articles which followed these discussions.[10] In large part, these discussions have treated the problem as a practical one, and a number of solutions have proposed. The TEI Guidelines summarize the basic options:

- concurrent markup, in which each hierarchy is encoded as a separate structure;[11]
- empty elements, in which an element is used to mark a point in the text;
- fragmentation, in which elements are broken into smaller sections which do not overlap.

Of these, concurrent markup best preserves the actual structuring of the source, but unfortunately it is not accommodated by existing SGML parsers. Each of the other two has its limitations, but lends itself to a particular subset of the encoding problems which the WWP most frequently encounters.

---

[10] See, for instance, Barnard, 1988.

[11] This is codified in SGML in the CONCUR feature; see Goldfarb, 1990, p. 177.

Using empty elements—elements which do not enclose any content, but simply mark a point in the text—avoids overlapping elements by not marking the element itself. There are two ways of using empty elements for this purpose. The first is as a "milestone", which marks the boundary between two regions of text. The use of such a boundary marker relies on the assumption that the elements it defines abut directly, without any interstitial material; where one element ends, the next one begins. This makes milestone elements ideal for encoding structures like pages or typographical lines: since each set of milestones defines a system of objects which follow one another without interruption, which never nest inside one another, and which together encompass the entire text (so that there is, for instance, nothing which is not on a page), the lack of explicitly enclosed elements does not create significant processing problems. Another way of using empty elements is as substitutes for begin-tags and end-tags; in effect, one empty element is defined as a "start" element, and its complementary element serves as a "close" element. Together, they enclose a textual feature just as ordinary tags do, but since they have no formal relationship to one another as far as an SGML parser is concerned, the area they mark can be enclosed within other elements without the risk of overlap. This latter variation lends itself to encoding things like quotations or other marked regions of text which span some other textual feature such as a paragraph or column. In this case, however, as the TEI Guidelines point out, the lack of a formal relationship between the enclosing elements can create some difficulties: extreme care must be taken to ensure that every start element is matched by a corresponding close element, and that they are explicitly and accurately linked together using attribute values. Otherwise there is no way for the processing software to know which pairs of elements constitute the boundaries for a given textual feature, as in the following example:

```
<highlightOpen>This text is highlighted in both red
and <highlightOpen>green ink,
but it might be unclear whether the second color is between
the two inner elements<highlightClose> or between
the second and fourth.<highlightClose>
```

In order to make this approach work, we need to use attribute values to specify the relationship between the pairs of tags (a relationship which, in ordinary SGML, would be explicit by virtue of one being a start-tag and the other an end-tag). In the following example, attributes link the first `<highlightOpen>` element with the second `<highlightClose>` element,

making it clear that the first color extends to the end of the selection, with the second lying only between the two middle elements:

```
<highlightOpen ID=HO1>This text is highlighted in both red
and <highlightOpen ID=HO2>green ink,
but it might be unclear whether the second color is between
the two inner elements<highlightClose corresp=HO2> or between
the second and fourth.<highlightClose corresp=HO1>
```

It should be noted, though, that since an SGML parser will need to look at the attribute values in order to accurately reconstruct this passage, this approach is somewhat inconvenient.

The WWP uses milestone elements in the way just described to encode features specifically pertaining to the physical structure of the source document. These include primarily pages and typographical lines, about both of which it can safely be assumed that when one line or page has ended, another one will begin immediately. Thus using a simple boundary marker for these features does not create any active difficulties. Using a boundary marker as a universal solution to the problem of overlapping elements, however, is undesirable. In some cases, both overlapping elements may be of a type which is inconvenient to encode using an empty boundary element. For instance, both quotations and paragraphs occur frequently enough and are of enough structural importance that they require tagging as normal elements; to tag them with empty elements throughout a text would generate unnecessary and awkward complexity.

For such a case, the second remaining approach suggested by TEI—fragmentation and joining—is more useful. This approach does not use empty elements, but instead divides one of the textual features into smaller segments which nest within the elements of the other textual feature, thus avoiding the overlap. For instance,

```
<sentence><quote ID=Q1 next=Q2>This
quotation,</quote> she said, <quote ID=Q2 prev=Q1>is
interrupted; it has been divided into two sections and linked
using attributes</quote>; this fact gave her great
satisfaction.</sentence>
```

The separate sub-elements are linked by using attribute values to identify them as part of the same feature. This approach retains the ability to validate the document, since textual features are still being recorded as elements, albeit smaller ones; what is really sacrificed is the easy identification of the entire feature as a single thing. This information is retained in the

attribute values, but may be more difficult for processing software to access and handle.

The WWP uses this fragmentation approach for a variety of purposes where empty boundary elements are inadequate: for instance, to encode interrupted quotations, or quotations which span several verse lines. However, there are some cases where neither of the approaches just described offers a truly satisfying solution. One particularly knotty problem arises for us in the case of serial publications which contain installments of different literary "works" spread out over a period of a few years. One such publication, *The Gleaner* by Judith Sargent Murray, comprises 100 sections—issues of Murray's column in the Massachusetts Magazine, together with other writings of hers—bound and published together in three volumes. The sections in the collection include installments of various serial works: a play, a novel, a sequence of essays on a particular topic. The works themselves are clearly of interest in their entirety; the user of the textbase needs to be able to reconstruct each one from its parts. At the same time, Murray's sequencing of the sections—the juxtapositions of different genres and topic—is also of interest and it may be crucial to researchers to be able to treat each volume or each section as a separate and integral object of study. These are considerations from the user's viewpoint; from the encoder's position there are further complexities. Each literary work within the collection is divided both into serial parts and into the divisions appropriate to its genre: acts and scenes, chapters, and the like. In many cases these divisions correspond to the boundaries of the issues, but in some cases (for instance, the novel *The Story of Margaretta*) the chapters may not even be in consecutive issues. In other serial works, where one wishes to encode the entire periodical, the task of arriving at sensible divisions of this text may be even more difficult. If one attempts to encode each scene of a play, for instance, as a division, then how is one to treat the intervening material (essays, editorials, poetry, etc.)? Even if each of these intervening items is also encoded as a textual division, the larger problem of how to identify the parts which belong to the higher-level division (the act) remains.

This is a problem which cannot be solved satisfactorily by using either empty boundary elements or linked segments. There are multiple overlapping hierarchies—the various structures of the separate "works" as well as that of the physical document—and at the very least all of the former have an equal claim to the researcher's attention, hence requiring effective treatment as elements. The WWP encodes this text using boundary elements

for the physical structure, and linked segments for the various parts of the literary works, with attributes identifying the parts and allowing them to be reconstructed. The <join> element provided by TEI enables the encoder to explicitly link all the parts of a particular textual structure, making it possible for processing software to treat them as a group. This solution does at least provide the necessary functionality. Its practical drawbacks are that from the encoding standpoint it is difficult to get right, and from the processing standpoint it requires more complicated treatment to reconstruct the structure of the text. This solution also leaves us with a document that has little internal structure at all; few of its important constitutive parts are accessible as integral units without active reconstruction, and none retains the important hierarchical ordering which is the real advantage of using SGML in the first place. At the practical level, we can work around these deficiencies, but the solution is inelegant. Without the implementation of the concurrent hierarchies feature of SGML, this problem remains mostly unsolved.

2.3. Further Issues

Multiple, overlapping hierarchies are everywhere, and a project attempting to encode for a wide-ranging audience will be especially likely to encounter them as a transcription issue. In its *CONCUR* feature, SGML itself offers a way of handling them, but in the absence of suitable software concurrent structures remain impractical to implement. Thus in practice one structure will always be privileged as the governing architecture of the document, while the other structures are marked in less explicitly hierarchical ways. Given the proper processing to reconstruct these latter features, from the user's point of view it may make little or no difference which structure is chosen as primary; the data can be delivered regardless of these more abstract considerations. The choice, then, should also address the more intangible question of how the encoding of the data should express the intellectual commitments of the encoding project, or the methodological or explanatory salience of a given structure. One must also arrive at a way of encoding the deprivileged hierarchy so as to retain as much as possible of its integrity. For the WWP, the structures which most frequently overlap are those of the physical book and the linguistic text, and we tend to privilege the latter. This is partly because of the research and usage needs of the preponderance of our audience. It is also partly because our ability to describe other physical

features of the text (ornamentation, typography, etc.) in detail is limited by our resources; knowing that research on these features will need to rely on a facsimile or the actual book in any case, we can focus our attention on the features that we can encode efficiently and effectively. However, we feel the importance of the physical book as the material medium through which the text circulated in its culture, and thus to the extent that we can, we preserve the physical structures of the book—signatures, pages, line breaks, catchwords, many details of page layout—as integrally as possible, using the methods described above.

Conclusion

In both of the cases discussed above, it is clear that the principles of SGML are intimately bound up with the WWP's conceptualization and solution of various transcription problems. This is to say that SGML and the TEI's implementation of it, quite apart from being either a help or a hindrance in *solving* a particular transcription issue, are of great use in thinking intelligently about it. It may sometimes seem that the particular formulations enforced by these systems create unnecessary complexity; however, in almost all cases this complexity is already latent in the document or the activity of transcription. What appear to be simple, natural systems (like pages with text on them) reveal their complexity when we attempt to map out their real structures in an explicit way; we see them as simple only because they are written deeply into our cultural systems. As a way of bringing such systems to the level of awareness, SGML is invaluable; we only need to remain aware of the structures it in turn creates before they too become naturalized and invisible.

Bibliography

ASKEW, (A.): 1563, "The Two Examinations of the worthy servant of God...", *Actes and Monuments*, ed. John Foxe (London: John Day), pp. 669–681.

BARNARD, (D.) *et al.*: 1988, "SGML-Based Markup: Problems and Solutions", *Computers and the Humanities*, 22, 265–76.

GOLDFARB, (C.): 1990, *The SGML Handbook* (Oxford: Clarendon Press).

IDE, (N.): May, 1991, "The Relevance of Computational Linguistics to Textual Studies", *Computers & Texts*, 5–7.

MCKENZIE, (D. F.): 1981, "Typography and Meaning: The Case of William Congreve", *The Book and the Book Trade in Eighteenth-Century Europe* (Hamburg: Dr. Ernst Hauswedell and Co.).

SPERBERG-MCQUEEN, (C.M.) and BURNARD (L.), eds.: 1994, *Guidelines for Electronic Text Encoding and Interchange (TEI P3)* (Oxford: Text Encoding Initiative).

RENEAR, (A.) *et al.*: forthcoming, "Refining Our Notion of What Text Really Is: The Problem of Overlapping Hierarchies", *Research in Humanities Computing*, ed. Nancy Ide and Susan Hockey (Oxford: Oxford University Press).