

## TACT and SGML together at last: *sgml2tdb*

*John Bradley*

---

Information Commons, 4 Bancroft Ave, Room 202, University of Toronto, Toronto, Ontario, Canada M5S 1A1

KEYWORDS: TACT, SGML, TEI

AFFILIATION: University of Toronto

E-MAIL: john.bradley@utoronto.ca

FAX NUMBER: +1 (416)978-6110

PHONE NUMBER: +1 (416)978-3995

As someone who is both the originator of TACT and still interested in computing in the humanities, I believe it is important to acknowledge the growing importance of SGML to our community. SGML was originally invented to ensure that new electronic texts destined for traditional publication would conform to a strict set of structural rules. Applying SGML to the editorial role of preserving complex, often ambiguous, structures found in pre-existing texts required a stretching of it well beyond what its designers originally intended. Nonetheless, as a result of the work of the Text Encoding Initiative (TEI), culminating in the publication of the P3 tag set, SGML has found an important new role as a standard markup scheme for scholarly electronic textual editions. Indeed, one oft-stated benefit of SGML to the publishing community – that text once marked up in SGML would continue to be useful for many years to come – has perhaps been most attractive to humanities scholars.

Most of SGML's technical development in the past resulted in tools which supported the construction of SGML documents, and their preparation for printing. More recently, the recently approved DSSSL (Document Style Semantics and Specification Language) ISO standard defines a mechanism for transforming SGML text into presentation text. OpenText's PAT software is the first text searching product that could be of broad interest to the humanities that actually (almost) uses real SGML tags.

Although I don't expect TACT to last as long as texts marked up with SGML I have believed for some time it would be useful to allow SGML texts to be brought into the TACT system. I've been working on and off on software to do this for over a year, and I am finally ready to release a preliminary version of this software – called *sgml2tdb* – at the Bergen Conference. With the delivery of this

program I believe we will have public domain software which when coupled with TACT programs such as Usebase or TACTweb can begin to make SGML texts readily useful to a larger scholarly community.

My goal in developing *sgml2tdb* has been to support the translation of SGML documents into a standard TACT 2.x TDB. Since existing TACT programs are entirely unaware of SGML, some aspects of SGML markup get, as it were, left at the door – and the implications of this will, I expect, be more evident in my presentation. A much more ambitious goal would be to develop software that would allow SGML to somehow be represented inside the system as well. Nonetheless, the immediate goal is to produce a TACT 2.x TDB from SGML text and it is important to understand the most important similarities and differences between SGML and the TDB format.

### Separation of text and markup

In both SGML and the TDB format the text is clearly separated from the markup scheme. SGML conventions focus on how SGML markup should be structured but says, in fact, very little about the text which fits between the markup. Although a few researchers, such as Willard McCarty in his endeavours to adapt TACT markup schemes to his needs for marking up Ovid's *Metamorphoses*, have blurred the distinction by "marking up" the words themselves, in reality the TACT software deals with what it recognizes as "the markup" in a very different way from the words found in the text. By keeping the "text" and the "markup" so clearly distinct, I find both TACT and SGML to already be similar.

### Textual Characters

Although SGML by itself has relatively little to say about the actual text itself between the SGML markup, TACT needs to be able to attach some limited semantics to these codepoints – what are the letters, how are they sorted, what are the diacritical characters, etc. It is necessary to provide *sgml2tdb* with information to allow it to convert what it is processing into words, symbols, letters, etc. The TEI project has recognized the need for more information of this kind in their Writing System Declarations, and I will discuss how the WSD could be used to fill this gap.

### Formatting of Text

SGML and TACT are quite different in the way that they deal with the formatting of text. Within TACT text must often be presented on the screen. Of course, the actual layout of the text is then clearly of some importance, and when I designed TACT I assumed that to a large extent the inputted

text file would be preformatted – TACT when displaying the text could simply present what it found – lines endings and all. Thus, the display text, as it is stored in the TDB, is meant to be “preformatted”.

On the other hand, it is very clearly a part of the SGML philosophy to say nothing at all about formatting. Instead, a part of the job of any software that ultimately has to display an SGML document is to get display information elsewhere (such as from a separate style sheet document which tells the program how to display each element, but which is not a actually a part of SGML), and use as a guide the structural markup information it actually finds in the text plus the information in this style sheet document to determine how the text it must display should be formatted.

Clearly, even though SGML is not really about formatting text, text destined for display by other TACT programs must be formatted by the time it gets into the TACT TDB file. Information must be provided that *sgml2tdb* can use to prepare the text in this way.

### Tagging Paradigms

TACT provides good support for textual tagging. Unfortunately, the set of assumptions that come with SGML markup and TACT’s COCOA-like markup are significantly different, and perhaps surprisingly, both models seem to me to represent ways people think about a text’s formal structure. In TACT one can tag many different structural entities. TACT imposes no association between the tags, and assumes that there is a “value” for any particular structural entity at all times in the text. If, for example, the text included tagging for “speaking character” in a play, at any point in the text – even within, say, the prefatory sections before the spoken text of the play itself – there must be a value for the “speaking character”. Of course, it is possible for the coder to provide a tag value such as “–” to indicate that here no character is speaking.

In SGML markup, all tagging is done in terms of textual elements, which must nest into one or more hierarchies. Thus, a play may divide into acts, which divide in scenes, which divide into speeches. When marking up the text one doesn’t think of marking “speaking characters” – instead, if one wishes (and it is supported by the DTD for the text one is working with) one can associate an attribute to a speech element that identifies the character. In parts of the text where there is no spoken text, there will be no “speech element”, and, no possibility for a “speaking character” attribute for such an element.

Since TACT 2.x represents markup as independant, parallel objects with values, the hierarchial model that SGML uses must be translated into the

TACT system by *sgml2tdb*, and the program must be given information about how this translation should occur. As well as outlining how this translation would be specified, I will discuss some of the design implications that would follow from changing the TDB format so that an SGML-style hierarchical markup could be fully preserved within the database.

### Standardized *sgml2tdb* specification files

By now it should be clear that *sgml2tdb* needs more information than that contained in the text and the document type definition to be able to produce a TDB. This extra information is provided in a special language I have developed expressly for *sgml2tdb* which is somewhat similar to Klaus Harbo’s “Copenhagen SGML Tool” (CoST), although my language is aimed specifically at the needs of TDB creation. In my presentation I will discuss its broad outlines.

There is one striking observation that can be made about it. Most of what goes into it can be derived not from the syntax of the elements and attributes (which is, of course, defined already in the associated document type definition), but instead from the semantics associated with the DTD’s elements and their attributes. Thus, as long as the semantics for elements and attributes for standardized DTD’s (such as TEI) are as well defined as their syntax, it may well be possible to also develop a standardized *sgml2tdb* specification file that will be independant of any particular text. For any text for which there is a both standardized DTD and *sgml2tdb* specifications it would then be possible to immediately produce a standardized TDB as well – purely from the information that is already present. During my presentation I will discuss my own progress in testing this hypothesis, by discussing the development of an *sgml2tdb* specification file for the TEI-lite format.