TeX and SGML: A Recipe for Disaster?

Peter Flynn
University College
Cork, Ireland
Internet: pflynn@curia.ucc.ie

Abstract

The relationship between TEX and SGML (Standard Generalised Markup Language, ISO 8879) has often been uneasy, with adherents to one system or the other displaying symptoms remininscent of the religious wars popular between devotees of TEX and of word-processors.

SGML and TEX can in fact coexist successfully, provided features of one system are not expected of the other. This paper presents a pilot program to test one method of achieving such a cohabitation.

Introduction

For many years, SGML and its relationship with TEX has been a frequent topic of presentation and discussion. Network users who read the TEXhax digest and the Usenet newsgroup comp.text.tex will be familiar with the sometimes extensive cross-postings to the sgml-l mailing list and the comp.text.sgml newsgroup. Two extremes are apparent in the misunderstandings: that SGML is some kind of desktop publishing (DTP) system; and that TEX or are exclusively for structured documentation. Such problems highlight the lack of information about the design of either system, as available to the novice, but also reveal the capabilities and limitations of both systems.

In fact, there is a parlous level of understanding about both TEX and SGML even in the printing and publishing industry, where one would expect a more sophisticated degree of understanding: in this author's personal hearing, so-called experts from major publishing houses have criticised TEX's 'lack of fonts' and SGML's 'lack of font control'.

It is perhaps worth emphasising the difference at this stage, for the non-expert, in that TEX is a typographic system principally for the creation of beautiful books (Knuth, 1984) (but also other printed documents: it is intended for putting marks on paper) and SGML (Goldfarb, 1990) is the international standard for describing the structure of documents (intended for document storage and control, which could, of course, include typesetting as one of many possibilities).

Publishing: the view from outside

A recent article (Beard, 1993) quotes John Watson, London Editorial Director of Springer-Verlag:

> We can use LATEX files, which many of our authors of books or papers with complex maths find convenient, but if they need serious editing, it's so expensive we have to mark up hardcopy and send it back to the author to make the changes. *TEX and LATEX are only a stop-gap. SGML hasn't really reached our authors yet.* What's really needed is a WYSIWYG system that's as universal as TEX, preferably in the public domain so all our authors and freelances can use it, and easy for subject specialists to edit on screen. And of course the output should be Linotron- as well as Postscript-compatible. **(Emphasis added.)**

This view of the world expresses an attitude common in the publishing field, that editing TEX is difficult, that the nature of TEX is impermanent, and that the only goal of all writing is for it to be printed on paper. While SGML has indeed 'not reached our authors yet', that is hardly the fault of SGML, when editing systems for handling SGML are readily available for most platforms.

The speaker's desires are very laudable, however much one may agree or disagree with the implied benefits of WYSIWYG systems, in that the software should be universal, easy to use and in the public domain. The speaker's complaints, however, deserve further analysis.

Editing. The speaker seems here to be confusing two aspects of the technical editorial process: mathematics editing and copy editing (editing text for production), both of which have to date been perceived as matters for the specialist, as those who use TEX in a professional pre-production capacity with publishers as clients have long recognised.

In the confusion, sight has been lost of the fact that editing a file of TEX source code need be no more of a problem than editing any other kind of file, if an adequate macro structure is provided, and it is probably less of a problem the better structured the text is. If the publisher's authors are unable or unwilling to adhere to the very straightforward guidelines put out by most publishers, it would appear a little ingenuous to blame TEX for their deficiencies.

There are large numbers of literate and numerate graduates with sometimes extensive TEX experience: if (as seems to be implied) editing may now be entrusted to authors, a publisher has little excuse for not employing some of these graduates on non-specialist editorial work. It is, however, as unnerving to hear publishers so anxious to encourage authors to undertake pre-press editing as it would be to hear them encourage non-mathematicians to undertake mathematical editing: it is precisely because the authors do not normally possess the specialist knowledge to do this that the work is handled by in-house or contract editors. The mechanics of editing a TEX document are not especially difficult, given proficiently-written macros, and there are some crafty editor programs around to assist this task. Training courses in elementary TEX abound, so if a publisher is serious about cutting pre-press costs by using TEX, the way lies open.

The typographic skill resides in implementing the layout: taking the typographer's specifications and turning them into TEX macros to do the job, ideally leaving the author and subsequent editor with as little trouble as possible to get in the way of the creative spirit. The implementation of design is, however, increasingly being left to the author, who may understandably resent having to undertake what is usually seen as a task for the publisher, and who may be ill-equipped to perform this task (Fyffe, 1969), especially if a purely visual DTP system is being used.

Impermanence. TEX has been around for nearly 15 years, longer than any other DTP system, and quite long enough for the mantle of impermanence to be shrugged off: there is no other system which can claim anywhere near that level of stability and robustness. However, the present writer would be among the first to disclaim any pretensions on the part of TEX to being the final solution to a publisher's problems (although properly implemented it has no difficulty in seeing off the competition). It is difficult, however, to understand what TEX is supposed to be a stop-gap for, because the logical conclusion a reader might draw from the quotation above is that SGML is some kind of printing system, which it is not, although it can be used for that purpose (for example, in conjunction with something like TEX).

Printing as a goal. WYSIWYG TEX systems exist for both PCs and Macintosh platforms, if a user feels compelled to see type springing into existence prematurely. There are also similar editors for SGML, ranging from the simple to the sophisticated. The misconception seems to be that printing on paper is always going to be the goal of the writer and the publisher, but even if we accept this goal as the current requirement, there appears to be no reason why both TEX and SGML cannot be used together to achieve this.

The increasing importance being attached to hypertext systems, especially in academic publishing, is amply evidenced by the presentations at scholarly conferences, for example (Flynn, 1993) the recent meeting of the Association for Literary and Linguistic Computing and the Association for Computing and the Humanities. While paper publication will perhaps always be with us, alternative methods are of increasing importance, and systems such as SGML are acknowledged as providing a suitable vehicle for the transfer and storage of documents (Sperberg-McQueen and Burnard, 1990) requiring multiple presentations.

Software development. Before we leave this analysis, it is worth asking if publishers who are seeking an easy-to-use, widely-available, public-domain WYSIWYG-structured editor would be prepared to back their demands with funding for the development of such a system. Organisations such as the Free Software Foundation are well-placed to support and coordinate such an effort, and there are ample human resources (and considerable motivation) in the research and academic environment to achieve the target.

Document Type Disasters

The newcomer to SGML is often perplexed by the apparent complexity of even simple Document Type Definitions (DTDs, which specify how a document is structured). Although there are several excellent SGML editors on the market, many users are still editing SGML in a plain file editor with perhaps the use of macro key assignments to speed the use of tags and entity references. Worse, the task of getting the document printed in a typographic form for checking by proofreaders who are unfamiliar with SGML can present a daunting task without adequate software.

While we have said that such software is readily available, there are two inhibiting factors: cost and complexity. Although we are now beginning to see wordprocessor manufacturers take an interest in SGML (WordPerfect, for example), the impecunious researcher or student is still at a disadvantage, as WYSIWYG software for SGML is still expensive for an individual.

The problem of complexity is not easily solved: designing a document at the visual level of typography is already understood to be a specialist task in most cases, and designing a document structure, which is a purely conceptual task, without visual representation, is at a different level of abstraction. However, document structure design is not normally the province of the publisher's author, and should not affect the author's use of a structured-document editor, once the initial concept has been accepted.

Into print. The `comp.text.sgml` newsgroup repeatedly carries requests from intending users for details of available editing and printing software, which are usually answered rapidly with extensive details. The low level of SGML's public image (the 'quiet revolution' (Rubinsky, 1992)) indicates one possible reason why the system is still regarded with misgivings by some people.

There have been several attempts in the past to develop systems which would take an SGML instance and convert its text to a TEX or LATEX file for printing. The earliest appears to have been `Daphne`, developed in the mid 1980s by the *Deutsche Forschungsnetz* in Berlin, and the most recent is `gf` (`comp.text.sgml`, 4.6.1993) from Gary Houston in New Zealand (available from the Darmstadt `ftp` server). Several other programs exist, including some written in TEX itself, but the principal stumbling-block seems to be the desire to make the program read and parse the DTD so that the instance can be interpreted and converted accordingly.

A DTD contains information principally about the structure of the documents which conform to it, rather than about its visual appearance. (It is of course perfectly possible to encode details on visual appearance in SGML, but this is more the province of the analyst or historian, who wishes to preserve for posterity the exact visual nature of a document.) The DTD is used to ensure conformance, often by an editor while the document is being written or modified, or by a parser (a program which checks the syntax and conformity of an instance to its DTD). Given the easy availability of various versions of a formal SGML parser (`sgmls`, from various `ftp` archives), there seems to be little point in embedding that process again in a formatter. Indeed, one conversion system reported to this author takes the route of using `sgmls` output as its input.

Through all these systems, however, runs the thread that somewhere in the SGML being used must reside all the typographical material needed to make the conversion to TEX (or indeed any typographical system) a one-shot process. As has been pointed out, this implies that the author or writer using SGML to create the document must embed all the necessary typographical data in the instance. Yet this is entirely the opposite of the natural use of SGML, which is to describe document *structure* or *content*, not its appearance. Predicating typographic matters ties the instance to one particular form of appearance, which may be wholly irrelevant.

Style and content. One of TEX's strongest features is that of the style file, a collection of macros to implement a particular layout or format. In particular, where this uses some form of standardised naming for the macros, as with LATEX or `eplain`, the portability of the document is greatly enhanced. A single word changed in the `documentstyle` and the entire document can be re-typeset in an entirely different layout, with (usually) no further intervention.

The convergence of SGML and TEX for the purposes of typesetting brings two main advantages: the use of TEX's highly sophisticated typesetting engine and the formally parsed structure of the SGML instance. In such a union, those elements of the DTD which do have a visual implication would migrate to a macro file, in which specific coding for the visual appearance *of the current edition* could be inserted, and the SGML instance would migrate to a TEX or LATEX file which would use these macros.

In this way, we would avoid entirely the predication of form within the SGML: it becomes irrelevant for the author to have to be concerned with the typographic minutiæ of how the publication will look in print (although obviously a temporary palliative can be provided in the form of a WYSIWYG editor). We also avoid tying the instance to any one particular layout, thus enabling the republication (or other reuse) in a different form at a later date with a minimum of effort.

The most undemanding form of conversion is thus one where the appearance is completely unreferenced in the SGML encoding. This means that the publisher (or typesetter) has all the hooks on which to hang a typographic implementation, but is not restricted or compelled to use any particular one of them.

A pilot program: sgml2tex

The author's own pilot attempt at this form of conversion can be seen in the SGML2TeX program, available by anonymous ftp from curia.ucc.ie in pub/tex/sgml2tex.zip. This was developed in PCL (a language written explicitly for high speed development on the 80*n*86 chips): WEB should probably be the basis for a future version.

The program reads an SGML instance character by character, and converts all SGML tags into TEX-like control sequences, by removing the < and > delimiters and prepending '\start' or '\finish' to the tag name. Attributes are similarly treated, within the domain of the enclosing element, and with their value given in curly braces as a TEX macro argument. Entity references are converted to simple TEX control sequences of the same name.

The output from the program is a .tex file and a .sty file. The .tex file contains an '\input' of the .sty file at the start, and also a '\bye' at the end; otherwise it is merely a representation of the instance in a form digestible by TEX or LATEX. The .sty file contains a null definition of every element, attribute and entity encountered in the instance. Thus the fragment

```
prepend '<tt>&bsol;start</tt>'
```

becomes

```
prepend
'\startTT{}\bsol{}start\finishTT{}'
```

in the .tex file, with the following definitions in the .sty file:

```
\def\startTT{}
\def\finishTT{}
\def\bsol{}
```

All line-ends, multiple spaces and tabs in the instance are condensed to single space characters.

It must be made clear that this pilot is not a parser: it does not read any DTD and has no understanding of the SGML being processed, although a planned rudimentary configuration file will allow a small amount of control over the elimination of specific elements where no conversion is desired. There is also no capability yet for handling any degree of minimisation, so all markup must be complete and orthogonal (as many parsers and editors already have the capability to output such non-minimised SGML code, this should not cause any problems). As the DTD is not involved, the instance being converted must therefore also have passed the parsing stage: it is the user's responsibility to ensure that only validly-parsed instances are processed. Additionally, no attempt has been made to support scientific, mathematical or musical tagging, as this is outside the scope of the pilot.

As it stands, therefore, the output file is a valid TEX file, although trying to process it with null definitions in the .sty file would result in its being treated as a single gigantic paragraph. However, editing the .sty file enables arbitarily complex formatting to be implemented: the present document (http://curia.ucc.ie/tlh/curia/doc/achallc.html) is a simple example.

Conclusions

The pilot program certainly *is* a stop-gap, being severely limited: there are many other related areas where SGML design, editing, display and printing tools are still needed. There is still no portable and widespread public-domain dedicated SGML editor such as would encourage usage (although an SGML-sensitive modification for emacs exists and the interest of WordPerfect has been noted). Although SGML import is becoming available for some high-end DTP systems, migration and conversion tools are still at a formative stage.

One particular gap is highlighted by the need for a program to assist the user in building a DTD, with a graphical interface which would show the structure diagrammatically, so that permitted and prohibited constructs can be analysed, and a valid DTD generated.

SGML has now passed the phase of 'new product' and is on its way to greater acceptance, but the real disaster would be for it to become an isolated system, unrelated to other efforts in computing technology. This will only be avoided by the concerted efforts of users and intending users in demanding software which can bridge the gaps.

Bibliography

Beard J. "The art and craft of good science", *Personal Computer World*, page 350, June 1993.

Fyffe C. *Basic Copyfitting*, London: Studio Vista, page 60, 1969.

Goldfarb C. The SGML Handbook, OUP, 1990.

Knuth D.E. *The TEXbook*, Addison-Wesley, 1984.

Rubinsky Y. "The Quiet Revolution", keynote speech, SGML92 Conference, October 1992.

Sperberg-McQueen C.M. and Burnard L., (eds). *Guidelines for the Encoding and Interchange of Machine-Readable Texts*, Draft version 1.1, ACH/ACL/ALLC, Chicago & Oxford, 1990.