Migrating to XML: The Case of the GUST Bulletin Archive

Włodzimierz Bzyl
Instytut Matematyki, Uniwersytet Gdański
80-952 Gdańsk
ul. Wita Stwosza 57
Poland
matwb@univ.gda.pl

Tomasz Przechlewski
Wydział Zarzadzania, Uniwersytet Gdański
81-824 Sopot
ul. Armii Krajowej 119/121
Poland
tomasz@gnu.univ.gda.pl

Abstract

Ten years of experience with TEX publishing of the GUST bulletin shows that Knuth's dream of highly portable TEX files is apparently an illusion in practice. Over the last decade, articles in the GUST bulletin have used at least six major formats (LATEX 2.09, transitional LATEX+NFSS, LATEX 2_ε, plain-based *TUGboat*, Eplain, and ConTEXt), numerous macro packages, fonts, and graphic formats. Many old articles are typeset differently nowadays, and some even cause TEX errors.

This situation motivates the following question: how do we avoid the same problem in the future? As the World Wide Web is quickly becoming the mainstream both of publishing and of information exchange we argue for migration to XML—a Web compatible data format.

In the paper we examine a possible strategy for storing GUST articles in a custom XML format and publishing them with both TEX and XSLT/FO. Finally, the problems of converting the TEX files to XML and possibility of using TEX4ht—an authoring system for producing hypertext—are discussed.

1 Introduction

The dominant role played by the Web in information exchange in modern times has motivated publishers to make printed documents widely available on the Internet. It is now common that many publications are available on the Web only, or before they are printed on paper. Articles published in the GUST bulletin are available on the Web in PostScript and PDF. Unfortunately, these formats decrease document accessibility, searching and indexing by Web search engines. For broad accessibility to automated services, it is better to use XML as the format of such data. However, one issue with XML is that it is difficult to maintain the high quality presentation of TEX documents. This is caused by incompatibilities between browsers and incomplete or immature implementations of W3C Consortium standards.

We are optimistic that these issues will disappear in the near future, and believe that XML will become pervasive in the online environment. However, in our context, a key to the adoption of XML is the degree to which it can be integrated with existing TEXnologies.

In this paper we examine one strategy for storing GUST articles in a custom XML format and publishing them with *both* TEX and XSLT/FO. Also, the problems of converting the existing TEX files to XML and the possibility of using TEX4ht—an authoring system for producing hypertext—are discussed.

2 TEX/LATEX and Other Document Formats

When the authors started work with TEX (many years ago), there was only a choice between closed-source applications based on proprietary formats, or

TEX, for publishing significant documents. Nowadays, the choice is much wider, as XML-based solutions are based on open standards and supported by a huge number of free applications. We do not need to write the tools ourselves. Thus the strategy of reusing what is publicly available is key in our migration plan.

On the other hand it would be unwise to switch to XML as the only acceptable submission format, because it would force many authors to abandon their powerful TEX-based editing environments to which they are accustomed, just to submit texts to our bulletin. Following this route, we would more likely end up with a shortage of submissions. Thus, we are preparing a mixed strategy with both TEX and XML as accepted formats. Papers submitted in LATEX will ultimately be converted to XML as an archival or retrieval format. Presentation formats will be XHTML, with corresponding PDF generated by a variety of tools. The work-flow of documents in this proposed framework is depicted on Fig. 1.

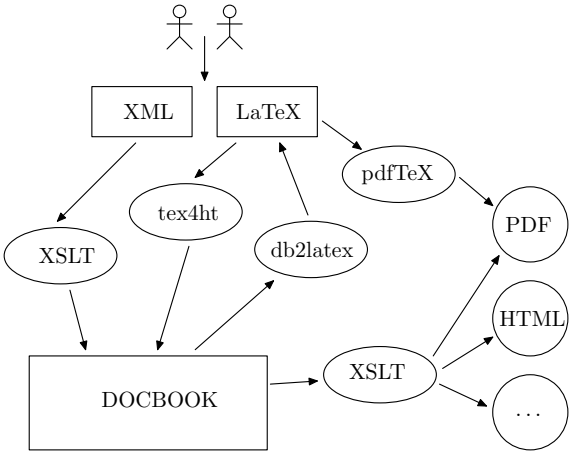

Figure 1: Processing diagram for XML/LATEX documents.

The XML implementation project described in the paper can be broadly divided into the following subtasks: DTD development, formatting development, and legacy information conversion [19]. We'll now describe these stages in detail.

3 DTD Development Considerations

There is no doubt (see for example [14, 19]) that the DTD development phase is of critical importance for the overall success of any SGML/XML project. Fortunately, thanks to the great interest in XML technology in recent years, there are several production-quality publicly available DTDs which could be adapted for our project. To make this choice, we preferred those which are widely used and for which formatting applications and tools are available. The following possible schemes were considered:

- DocBook [21], a complete publishing framework, i.e., schemes plus XSLT/DSSSL stylesheets for conversion to presentation formats; actively developed and maintained; the de facto standard of many Open Source projects; widely known and used.
- TEI [1], another complete, actively developed publishing framework. Not as popular as DocBook, used mainly in academia.
- The LATEX-based DTD developed in [7] (further referred as LWC DTD). The similarity to the structure of LATEX is an advantage of this DTD for our project.
- Others, such as DTD for GCA/Extreme conferences, X-DiMi from the Electronic Thesis and Dissertations Project, and the LATEX-like PMLA developed by one of the authors [15].

Industry-standard DTDs tend to be too big, too complex, and too general for practical use in specific cases (cf. [14, p. 29]). In particular, the DocBook and TEI DTDs seem to be too complex for marking-up documents conforming to LATEX format.

As a result, users frequently employ the technique of using different DTDs at different stages of the editorial process. Following Maler [14], we will call the DTD common to a group of users within an interest group as a *reference* DTD, while those used solely for editing purposes as an *authoring* DTD. Translation from one DTD to another may be easily performed with an XSLT stylesheet.

We decided to use a simplified LWC DTD as authoring DTD and DocBook as reference DTD. Providing a simple DTD should expand the group of prospective authors. For example, many GUST members are experts in typography or Web design but not necessarily TEX hackers.

The simplification consists of restricting the document hierarchy only to article-like documents, and removing back matter tags (`index`, `glossary`) and all presentation tags (`newline`, `hspace`, etc.). Also, the optional status of meta-data, for example the `title`, `abstract`, `keywords` tags, was changed to required. The resulting DTD contains 45 elements compared to 64 in the original one.

For better maintainability, we rewrote our version of LWC DTD into RNC syntax. The RNC schema was introduced by Clark [6], and recently adopted as an ISO standard. It has many advantages

over DTD or W3C Schema syntax, namely simplicity and an included documentation facility.[1]

As the structure of our documents is not particularly complex, it may be feasible to develop several authoring DTDs targeted at different groups of authors, for example one for those preferring ConTEXt-like documents, another for those used to GCA conference markup, etc., and then map those documents to the reference DTD with XSLT.

4 Formatting with XSLT

For presentation, LWC documents are first transformed to DocBook with a simple XSLT stylesheet.

The DocBook XSL stylesheets [22] translate an XML document to XHTML or FO [18]. As they are written in a modular fashion, they can be easily customized and localized. To publish XHTML from XML documents, an XSLT engine is needed such as Kay's `saxon` [11] or Veillard's `xsltproc` [20].

For hard copy output, a two-step process is used. First, the XSLT engine produces formatting objects (FO) which then must be processed with a formatting object processor for PDF output.[2] The detailed transformation work-flow is depicted in Fig. 2.

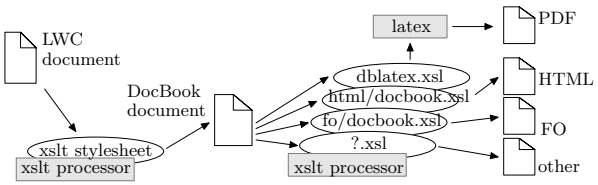

Figure 2: Processing details of LWC documents with XSLT/FO.

With just a few customizations the translation from XML to XHTML presents no obstacles (except for math formulas). On the other hand, the quality of the PDF produced with the publicly available `fop` processor from the Apache project is poor compared to that obtained with TEX.

Instead of generating FO objects one can use XSLT to translate XML directly to high-level LATEX. That is the method used in db2latex [3] (see also a clone project: dblatex/dbcontext [9]; the latter, of course, generates files processable by ConTEXt). The output can be customized at XSLT stylesheet level as well as by redefining appropriate LATEX style files. MathML markup is translated with XSLT to LATEX and supported natively.[3]

The translation from DocBook to LATEX implemented in these tools is incomplete. To get reasonable results, prior customization to local needs is required. The main advantage of this approach is that we use TEX—a robust and well known application.

5 The GUST Bulletin Archive

When considering the conversion of the GUST archive to XML we have two points in mind: first, we recognize the long-term benefits of an electronic archive of uniformly and generically marked-up documents; and second, to take the opportunity to test the whole framework using 'real' data.

During the last 10 years, 20 volumes of the GUST bulletin were published, containing more than 200 papers. From the very beginning GUST was tagged in a modified *TUGboat* style [2]. The total output is not particularly impressive, but the conversion of all articles to XML isn't a simple one-night job for a bored TEX hacker:

- they were produced over an entire decade and were written by over 100 different authors.
- they were processed with at least six major formats (LATEX 2.09, transitional LATEX+NFSS, LATEX 2_ε, plain-based *TUGboat*, Eplain, and finally ConTEXt), using numerous macro packages, fonts, and graphic formats.[4]

As a group, the GUST authors are not amateurs, producing naïve TEX code. On the contrary they are TEX experts, writing on a diverse range of subjects using non-standard fonts, packages and macros. For example, Fig. 3 shows the detailed distribution of the TEX formats used in GUST.

In total, there were 134 plain TEX articles, compared to 87 for LATEX. LATEX authors used 74 different packages, while those preferring plain TEX nology used 139 different style files. The proportion of other formats (Eplain, ConTEXt, BLUE) was insignificant (only a few submissions). It can also be noted from Fig. 3 that in recent years, the proportion of plain TEX submissions has decreased substantially in favor of LATEX.

It is obviously very difficult to maintain a repository containing papers requiring such a diverse

[1] It is possible to convert between different schema languages for XML with the `trang` program [5]. There is also a nxml-mode for GNU Emacs for editing of XML which features highlighting, indentation, and on the fly validation against an RNC schema [4].

[2] Modern browsers have XSLT engines built-in. So, it suffices to attach to a document appropriate stylesheets to make the transformation on the fly.

[3] One approach which we did not try is to format FO files with TEX. This method is implemented by S. Rahtz' Passive TEX [17].

[4] Needless to say, all of these packages have been evolving during the last 10 years, many of them becoming incompatible with each other.

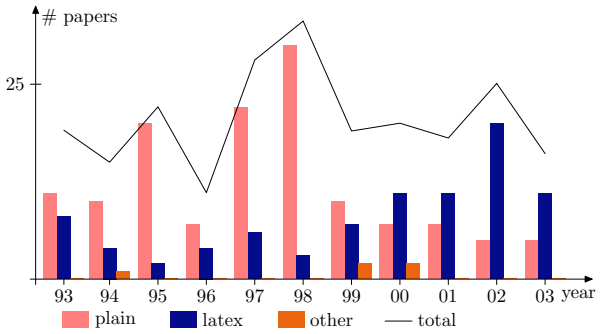

Figure 3: Distribution of TEX formats used by GUST authors.

range of external resources (macros, fonts). As a result, many old papers are now unable to be typeset owing to changes in underlying macros or fonts.

6 Conversion from TEX to XML

It may be surprising that only few papers report successful conversion from TEX to XML: Grim [8] describes successful large-scale conversion in a large academic institution, while Rahtz [16] and Key [12] describe translation to SGML at Elsevier.

Basically when converting TEX to XML the following three approaches have been adopted [16]:

- Perl/Python hackery combined with manual retyping and/or manual XML marking-up.
- Parsing TEX source files not with `tex`, but with another program which generates SGML/XML. This is the approach used by `ltx2x` [23], `tralics` [8] and `latex2html`,[5] which replace LATEX commands in a document by user-defined strings.
- Processing files with TEX and post-processing the DVI files to produce XML. This is the way `tex4ht` works.

Although the conversion performed with `tralics` is impressive, we found the application very poorly documented. After evaluation of the available tools and consulting the literature [7], we decided to use TEX4ht—a TEX-based authoring system for producing hypertext [10].

Because TEX formats contain many visually-oriented tags, we could not expect to automatically convert them to content-oriented XML markup.[6]

For example, the *TUGboat* format requires only the metadata elements title and author name(s); author address(es) and webpage(s) of the author(s) are often absent and there is no obligation for abstracts and keywords. Therefore, most of the GUST articles lack these valuable elements. Moreover, bibliographies are inconsistently encoded.[7]

Having said that, our plan is to markup as many elements as possible.

7 Translating TEX to XML with TEX4ht

Out of the box, the TEX4ht system is configured to translate from plain, LATEX, *TUGboat* (ltugboat, ltugproc), and Lecture Notes in Computer Science (llncs) formats to HTML, XHTML, DocBook, or TEI. To translate from, say, *TUGboat* to our custom XML format the system needs to be manually configured. Because the configuration of TEX4ht from scratch is a non-trivial task, we consider other more efficient possibilities.

The TEX4ht system consists of three parts: (1) Style files which enhance existing macros with features for the output format (HTML, XHTML, etc.).[8] (2) The `tex4ht` processor which extracts HTML or XHTML, or DocBook, or TEI files from DVI files produced by `tex`. (3) The `t4ht` processor which is responsible for translating DVI code fragments which need to be converted to pictures; for this task the processor uses tools available on the current platform.

As mentioned above, the conversion from a visual format to an information-oriented one cannot be done automatically. Let's illustrate this statement with the following example marked with plain TEX macros:[9]

```
\noindent {\bf exercise, left as an}
{\it adj\/} {\ss Tech} Used to complete
a proof when one doesn't mind a
{\bf handwave}, or to avoid one entirely.
The complete phrase is: {\it The proof
\rm(or \it the rest\/\rm) \it is left as an
exercise for the reader.\/} This comment
has occasionally been attached to unsolved
research problems by authors possessed of
either an evil sense of humor or a vast
faith in the capabilities of their
audiences.\hangindent=1em
```

After translation of this fragment to XHTML by `tex4ht` we obtain:

```
<p class="noindent"><span
class="cmbx-10">exercise, left as an
</span><span class="cmti-10">adj
```

[5] `latex2html` was not considered as its output is limited to HTML.

[6] For example, see [16, 8]. Other examples, based on GUST articles, are presented below.

[7] Publicly available tools (see [13] for example) can automatically mark up manually keyed bibliographies.

[8] Altogether over 2.5M lines of TEX code. Compare this with 1M code of the LATEX base macros.

[9] The text comes from "The Project Gutenberg Etext of The Jargon File", Version 4.0.0.

```
</span><span class="cmss-10">Tech
</span>Used to complete a proof when one
doesn&#x2019;t mind a <span
class="cmbx-10">handwave</span>, or to
avoid one entirely. The complete phrase
is: <span class="cmti-10">The proof
</span>(or <span class="cmti-10">the
rest</span>) <span class="cmti-10">is left
as an exercise for the reader.
</span>This comment has occasionally been
attached to unsolved research problems by
authors possessed of either an evil sense
of humor or a vast faith in the capabilities
of their audiences.</p>
```

and this could be rendered by a browser as:

exercise, left as an *adj* Tech Used to complete a proof when one doesn't mind a **handwave**, or to avoid one entirely. The complete phrase is: *The proof* (or *the rest*) *is left as an exercise for the reader.* This comment has occasionally been attached to unsolved research problems by authors possessed of either an evil sense of humor or a vast faith in the capabilities of their audiences.

We can see that `tex4ht` uses 'span' elements to mark up font changes. These visual tags could be easily remapped to logical ones unless fragments of text with different meaning are marked with the same tag. Here the tag `cmti-10` was used to tag both the short form 'adj' and the example phrase (shown in the green italic font). To tag them differently we need different TEX macros specially configured for TEX4ht. Note that the `\hangindent=1em` was ignored by `tex4ht`. This command could not be simulated, because hanging indentation is not supported by browsers.

So, the markup produced by the `tex4ht` program is not logical markup. To get logical markup the GUST format should be reworked and reconfigured for TEX4ht.

Instead of configuring TEX4ht we could use an XSLT stylesheet to remap elements referencing XML format. This could be an easier route than configuring the system from scratch, while some TEX4ht configuration could also help. So, a combination of the two methods is envisaged to provide the best results.

8 Conclusion and Future Work

We have not completed the conversion yet. However, based on the experience gained so far we can estimate that almost 70% of the whole archive should be converted with little manual intervention. Semi-automatic conversion of another 15% (34 papers) is possible, with prior extensive changes in markup. Conversion of remaining 15% is impossible or useless, where 'impossible' means the paper is easier to retype than try to recompile and adjust `tex4ht` just for a particular single case, and 'useless' applies to papers demonstrating complicated graphical layouts, or advanced typesetting capabilities of TEX.

Although our system needs improvement—conversion of math is the most important remaining item to investigate—we are determined to start to use it in a production environment.

Finally, we note that many of our conclusions and methods are also applicable to *TUGboat*, because the format used for typesetting GUST bulletin differs only slightly from the one used for *TUGboat*.

References

[1] Lou Burnard and C. M. Sperberg-McQueen. TEI lite: An introduction to text encoding for interchange. `http://www.tei-c.org/Lite/`, 2002.

[2] Włodek Bzyl and Tomasz Przechlewski. An application of literate programming: creating a format for the Bulletin of the Polish TUG. *TUGboat*, 14(3):296–299, October 1993.

[3] Ramon Casellas and James Devenish. `DB2LaTeX` XSL stylesheets. `http://db2latex.sourceforge.net`, 2004.

[4] James Clark. NXML mode for the GNU Emacs editor. `http://www.thaiopensource.com/download`, 2003.

[5] James Clark. Trang—multi-format schema converter based on RELAX NG. `http://www.thaiopensource.com/relaxng/trang.html`, 2003.

[6] James Clark and Makoto Murata. Relax NG specification. `http://www.relaxng.org/`, 2001.

[7] Michel Goossens and Sebastian Rahtz. *LATEX Web Companion*. Addison-Wesley, 2001.

[8] Jose Grim. Tralics. In *EuroTEX Preprints*, pages 38–49, 2003. `http://www-sop.inria.fr/miaou/tralics`. Final version to appear in *TUGboat*.

[9] Benoît Guillon. DocBook to LATEX/ConTEXt publishing. `http://dblatex.sourceforge.net`, 2004.

[10] Eitan Gurari. `tex4ht`: LATEX and TEX for hypertext. `http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn.html`, 2004.

[11] Michael Kay. SAXON—the XSLT and XQuery Processor. `http://saxon.sourceforge.net`, 2003.

[12] Martin Key. Theory into practice: working with SGML, PDF and LATEX. *Baskerville*,

5(2), 1995. ftp://tug.ctan.org/pub/tex/usergrps/uktug/baskervi/5_2/.

[13] Language Technology Group. LT TTT version 1.0. http://www.ltg.ed.ac.uk/software/ttt, 1999.

[14] Eve Maler and Jeanne El Andaloussi. *Developing SGML DTDs: From Text to Model to Markup*. Prentice Hall PTR, 1995.

[15] Tomasz Przechlewski. Definicja dokumentu typu PMLA. http://gnu.univ.gda.pl/~tomasz/sgml/pmla/, 2002.

[16] Sebastian Rahtz. Another look at LaTeX to SGML conversion. *TUGboat*, 16(3):315–324, September 1995. http://www.tug.org/TUGboat/Articles/tb16-3/tb48raht.pdf.

[17] Sebastian Rahtz. PassiveTeX. http://www.tei-c.org.uk/Software/passivetex, 2003.

[18] Robert Stayton. *Using the DocBook XSL Stylesheets*. Sagehill Enterprises, http://www.sagehill.net/docbookxsl/index.html, 2003.

[19] Brian E. Travis and Dale C. Waldt. *The SGML Implementation Guide: A Blueprint for SGML Migration*. Springer-Verlag, 1996.

[20] Daniel Veillard. LIBXSLT—the XSLT C library for Gnome. http://xmlsoft.org/XSLT, 2003.

[21] Norman Walsh and Leonard Muelner. *DocBook: The Definitive Guide*. O'Reilly, 1999. http://www.docbook.org/tdg/en/html/docbook.html.

[22] Wiki. DocBook XSL Stylesheets. http://docbook.org/wiki/moin.cgi/DocBookXslStylesheets, 2004.

[23] Peter R. Wilson. LTX2X: A LaTeX to X Auto-tagger. http://www.ctan.org/tex-archive/support/ltx2x, 1999.