The TEI/TEX Interface

Sebastian Rahtz

March 2005

Abstract

In the view of many people, the natural way to prepare a typeset document is to use \LaTeX{} or Con\TeX{}. It produces high-quality output, the source document is a clean mixture of text and markup, and it works on any computer. For another group of people, however, the natural way to proceed is to prepare a validated XML document which can be used to either make a web page or to make a printed document. One choice of an XML encoding for this group is the Text Encoding Initiative (TEI) scheme.

This paper is in two parts. The first part examines the arguments for and against authoring in XML, rather than \TeX{}, and demonstrates how some common \TeX{} situations are catered for in TEI XML.

The second part of the paper examines how, if we do choose XML, we can continue to harness the power of \TeX{}. We examine the four main routes of

- using a modified \TeX{} to read XML directly;
- translating XML direct to high-level \TeX{};
- translating our XML to another XML which is functionally identical to \LaTeX{} and then translating that; and
- translating XML to an XML-based page description language (XSL FO), and processing that with \TeX{}.

None of these is completely satisfactory, and we end by considering what hope there is for the future.

1 The TEI / \TeX{} interface

musings and reports

2 Personal background

I am Sebastian Rahtz:

- Information Manager for Oxford University Computing Services
- Manager of OSS Watch, the UK national Open Source Advisory Service
- Oxford representative on the Board of Directors of the Text Encoding Initiative Consortium; member of the TEI Technical Council, and convenor of its Meta Language working party
- Long-time (coming up to 20 years) \TeX{} sorcerer (using the classification of Ursula Le Guin, not Don Knuth)
- Overall editor of \TeX{}live
3 TEI Background

The TEI

- an international and interdisciplinary standard that helps libraries, museums, publishers, and individual scholars represent all kinds of literary and linguistic texts for online research and teaching
- a comprehensive and well-documented markup language for all kinds of text material, from manuscripts to dictionaries, from film scripts to web pages.
- An XML vocabulary, coming up to a new release (P5) using XML schema languages

4 Example, part 1

```xml
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>The TEI/TeX interface</title>
        <author>Sebastian Rahtz</author>
      </titleStmt>
      <editionStmt>
        <edition>
          <date>March 2005</date>
        </edition>
      </editionStmt>
    </fileDesc>
    <revisionDesc>
      <change>
        <date>Date: 2005/03/10</date>
        <respStmt>
          <name>Author: rahtz</name>
        </respStmt>
        <item>Revision: #1</item>
      </change>
    </revisionDesc>
  </teiHeader>
</TEI>
```

5 Example, part 2

```xml
<text>
  <body>
    <div>
      <head>Personal background</head>
      <p>I am <hi>Sebastian Rahtz</hi>:</p>
      <list>
        <item>Information Manager for <emph>Oxford University Computing</emph> Services</item>
        <item>Manager of <emph>OSS Watch</emph>, the UK national Open Source Advisory Service</item>
        <item>Oxford representative on the Board of Directors of the <emph>Text Encoding Initiative Consortium</emph></item>
      </list>
    </div>
  </body>
</text>
```
6 Example 3

Member of the TEI Technical Council, and convenor of its Meta Language working party

Long-time (coming up to 20 years) TeX sorcerer

<note>Using the classification of Ursula Le Guin, not Don Knuth or J K Rowling</note>

---

Example 3

TUT01 Proceedings EuroTeX2005 – Pont-à-Mousson, France

40 The TEI/TEX Interface

Sebastian Rahtz
7 The TEI world view

8 The \LaTeX{} world view
9 Core differences between TEI XML and (eg) L\(\text{ATEX}\)

<table>
<thead>
<tr>
<th>TEI XML</th>
<th>L(\text{ATEX})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Markup using (most of) Unicode</td>
<td>Markup using ASCII (extensible with difficulty)</td>
</tr>
<tr>
<td>Verbose but consistent</td>
<td>Concise but arbitrary</td>
</tr>
<tr>
<td>International standard for markup</td>
<td>Private extensible markup</td>
</tr>
<tr>
<td>Unicode character encoding</td>
<td>Variable character encoding</td>
</tr>
<tr>
<td>Single syntax</td>
<td>Syntax determined by application</td>
</tr>
<tr>
<td>Vocabulary choice constrained by schema</td>
<td>Vocabulary dynamically extensible and changeable</td>
</tr>
<tr>
<td>Vocabulary checkable</td>
<td>Vocabulary only constrained by syntax</td>
</tr>
<tr>
<td>Language separate from processing</td>
<td>Processor and language intermixed</td>
</tr>
<tr>
<td>—</td>
<td>Builtin math engine</td>
</tr>
<tr>
<td>—</td>
<td>Builtin tabular engine</td>
</tr>
<tr>
<td>Multiple processors</td>
<td>One reliable processor</td>
</tr>
</tbody>
</table>

10 Why markup schemas?

So we want machine readable text:

- We need a reasonable notation (XML or \(\text{TEX}\))
- We need a character encoding system (Unicode)
- We need to make up vocabularies
- We need to be able to process our texts

What influences our choice? We

- want to interchange texts and tools with others
- need to have a formal way to express conditions about our markup
- should find a place to document our vocabulary

11 What we might do with a schema
12 Layers, using an XML schema

1. vocabulary
   
   <list>, <item>, <label>

2. suggested usage rules
   
   element list { item+ }

3. constraints on text
   
   figure.attributes.url.content = xsd:anyURI

4. dependency rules (project specific)
   
   <if test="self::list[@type='gloss'] and not(child::label)">
   
   <message>gloss lists must have <label> children</message>
   
   </if>

5. lookup rules
   
   <if test="document('lookup.xml')/people/person[@id=current::@ref]">
   
   <message>this person does not exist in the database</message>
   
   </if>

6. common sense rules
   
   Don’t use table markup to force layout

13 Does the TEI cover all these?

As of today, the TEI Guidelines contain:

vocabulary 362 elements, 95 attributes, 88 classes

suggested usage rules 24 modules with 7185 lines of rules in compact Relax NG

constraints on text W3C Schema datatyping

dependency rules none formally expressed yet

lookup rules unimplemented, but have a place

common sense rules a lot of descriptive prose

A good start.
14 An editors view

15 Example 3

Roma: generating validators for the TEI

Search TEI database:

Submit Query

Text Encoding Initiative

Warning! this version of Roma uses a pre-release draft of TEI P5 from 1st October 2004

Modules

List of TEI Modules

<table>
<thead>
<tr>
<th>Module name</th>
<th>A short description</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>add analysis</td>
<td>Simple analytic mechanisms</td>
<td></td>
</tr>
<tr>
<td>add certainty</td>
<td>Certainty and uncertainty</td>
<td></td>
</tr>
<tr>
<td>add core</td>
<td>Elements available in all forms of the TEI main DTD</td>
<td></td>
</tr>
<tr>
<td>add corpus</td>
<td>Header extensions for Corpus Texts</td>
<td></td>
</tr>
<tr>
<td>add declarats</td>
<td>Feature System Declaration</td>
<td></td>
</tr>
<tr>
<td>add dictionaries</td>
<td>Base tag set for printed dictionaries</td>
<td></td>
</tr>
<tr>
<td>add drama</td>
<td>Base tag set for Performance texts</td>
<td></td>
</tr>
<tr>
<td>add figures</td>
<td>Tables, Formulas, Figures</td>
<td></td>
</tr>
<tr>
<td>add gajli</td>
<td>Character and Glyph documentation</td>
<td></td>
</tr>
<tr>
<td>add header</td>
<td>The TEI Header</td>
<td></td>
</tr>
<tr>
<td>add iso-646</td>
<td>Feature Structures</td>
<td></td>
</tr>
<tr>
<td>add linking</td>
<td>Linking, Segmentation and Alignment</td>
<td></td>
</tr>
<tr>
<td>add metadescription</td>
<td>Manuscript Description</td>
<td></td>
</tr>
<tr>
<td>add namedates</td>
<td>Additional classes for names and dates</td>
<td></td>
</tr>
<tr>
<td>add nets</td>
<td>Graphs, networks and trees</td>
<td></td>
</tr>
<tr>
<td>add shareheader</td>
<td>Auxiliary DTD for Independent Header</td>
<td></td>
</tr>
<tr>
<td>add spoken</td>
<td>Base tag set for Transcribed Speech</td>
<td></td>
</tr>
<tr>
<td>add tagdocs</td>
<td>Declares the elements making up the module documentation module</td>
<td></td>
</tr>
<tr>
<td>add tel</td>
<td>Main document type declaration file</td>
<td></td>
</tr>
<tr>
<td>add textcrit</td>
<td>Tags for text criticism</td>
<td></td>
</tr>
</tbody>
</table>

List of selected Modules

<table>
<thead>
<tr>
<th>Module name</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>remove core</td>
<td></td>
</tr>
<tr>
<td>remove tel</td>
<td></td>
</tr>
<tr>
<td>remove header</td>
<td></td>
</tr>
<tr>
<td>remove textstructure</td>
<td></td>
</tr>
</tbody>
</table>
16 Another graphical view of TEI and \TeX

\begin{center}
\begin{tikzpicture}
  \node[rectangle, draw, inner sep=1cm, text width=20cm] (in) {
    Semantic markup, presentation markup, style, typesetting engine, conversion engine
  };
  \node[rectangle, draw, inner sep=1cm, text width=5cm, below of=in, yshift=-3cm] (xml) {XML};
  \node[rectangle, draw, inner sep=1cm, text width=5cm, right of=xml, xshift=3cm] (schema) {Markup};
  \node[rectangle, draw, inner sep=1cm, text width=5cm, right of=schema, xshift=3cm] (transformation) {Transformation};
  \node[rectangle, draw, inner sep=1cm, text width=5cm, below of=transformation, yshift=-3cm] (typesetting) {Typesetting};
  \node[rectangle, draw, inner sep=1cm, text width=5cm, below of=transformation, yshift=-3cm] (database) {Database};
  \node[rectangle, draw, inner sep=1cm, text width=5cm, right of=typesetting, xshift=3cm] (unicode) {Unicode};
  \node[rectangle, draw, inner sep=1cm, text width=5cm, right of=database, xshift=3cm] (schema) {Schema};
  \node[rectangle, draw, inner sep=1cm, text width=5cm, right of=schema, xshift=3cm] (transformation) {Schema};
  \node[rectangle, draw, inner sep=1cm, text width=20cm, below of=transformation, yshift=-3cm] (output) {Out};
  \node[rectangle, draw, inner sep=1cm, text width=20cm, above of=in, yshift=-3cm] (input) {In};
  \draw[->] (in) -- (xml);
  \draw[->] (xml) -- (schema);
  \draw[->] (schema) -- (transformation);
  \draw[->] (transformation) -- (typesetting);
  \draw[->] (typesetting) -- (database);
  \draw[->] (database) -- (unicode);
  \draw[->] (unicode) -- (input);
\end{tikzpicture}
\end{center}

17 Using \TeX\ behind XML

1. using a modified \TeX\ to read XML directly;
2. translating XML direct to high-level \TeX;
3. translating our XML to another XML which is functionally identical to L\LaTeX\ and then translating that; and
4. translating XML to an XML-based page description language (XSL FO), and processing that with \TeX\ (XSLFO).

17.1 \TeX\ reads XML directly: Con\TeX\t

Using mapping files:

\begin{verbatim}
\defineXMLEnvironment[article][id=\undefined]
  {\XMLDBpushelement\currentXMLElement
   \XMLDBmaystartdocument
   \XMLDBmayensurebodymatter}
  {\XMLDBmayfinishdocument
   \XMLDBpopelement}

\defineXMLEnvironment[itemizedlist]
  {\XMLDBpushelement\currentXMLElement \XMLDBmayensurebodymatter
   \doifsamestringelse{\XMLPar{itemizedlist}{spacing}{normal}{compact}
      {\startitemize[packed]}
      {\startitemize}
      \defineXMLIgnore[titleabbrev]%
    \defineXMLEnvironment[titleabbrev]
      {\item\XMLDBcontinuepartrue\ignorespaces}{}%
  }
  {\stopitemize\XMLDBpopelement}
\end{verbatim}
17.2 [1] \TeX reads XML directly: xml\TeX

\XMLelement{TEI.2}
{ \documentclass{article}
   \usepackage[bookmarks=false]{hyperref}
   \usepackage[teixml]
   \begin{document} 
   \end{document}
}

\XMLelement{ref}
{ \XMLattribute{target}{\reftarget}{} }
{ \xmlgrab }
{ \hyperref[\reftarget]{#1} }

(used for Passive\TeX XSL FO processor)

17.3 [1] \TeX reads XML directly: problems

1. gobbledygook, even by \TeX standards: only experts need apply
2. limited access to document tree
3. (xml\TeX) forced grouping makes mapping some constructs almost impossible
4. catcode issues in auxiliary files

17.4 [2] Translate XML to high-level \LaTeX

<xsl:template match="tei:list">
<xsl:choose>
   <xsl:when test="@type='gloss'">
      \begin{description}
         <xsl:apply-templates mode="gloss" select="tei:item"/>
      \end{description}
   </xsl:when>
   <xsl:when test="@type='unordered'">
      \begin{itemize}<xsl:apply-templates/>
      \end{itemize}
   </xsl:when>
   <xsl:when test="@type='ordered'">
      \begin{enumerate}<xsl:apply-templates/>
      \end{enumerate}
   </xsl:when>
   <xsl:otherwise>
      \begin{itemize}<xsl:apply-templates/>
      \end{itemize}
   </xsl:otherwise>
</xsl:choose>
</xsl:template>

17.4.1 (dirty details)

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{ucs}
17.5  [2] Translate XML to high-level \LaTeX: problems

1. Remaining catcode problems (\, {, })

2. When \LaTeX signals an error, where is it in the source?

3. Where do you make style decisions?
   
   • \tableofcontents or <divGen type="toc"/>
   • \section{Introduction} or \section{1. Introduction}
   • \def{xxxxx \def{yyyyy} or \usepackage{foo}

17.6  [3] Transform XML to XML-ised \LaTeX

Transform to

```xml
<cmd name="documentclass">
  <opt>12pt</opt>
  <parm>letter</parm>
</cmd>
<env name="document">
  <cmd name="author" nl2="1">
    <parm>A. U. Thor</parm>
  </cmd>
  <cmd name="title" nl2="1">
    <parm>A SHORT STORY</parm>
  </cmd>
  <cmd name="maketitle" nl2="1" gr="0"/>
  <cmd name="section" nl2="1">
    <parm>A SHORT STORY</parm>
  </cmd>
</env>
```

and thence to

```latex
\documentclass[12pt]{letter}
\begin{document}
\author{A. U. Thor}
\title{A SHORT STORY}
\end{document}
```

17.7  [3] Transform XML to XML-ised \LaTeX: ups and downs

✓ You don’t have to worry about \{ and \}

✗ It takes another processor

✗ It inserts yet another layer of obscurity between author and error on printout

✓ Allows for implementation using the first technique

```xml
<fo:list-block margin-right="10pt" space-before="6pt" space-after="6pt" margin-left="15pt">
  <fo:list-item space-before.optimum="4pt">
    <fo:list-item-label>
      <fo:block margin-right="2.5pt" text-align="center">
        &amp;#2219;
      </fo:block>
    </fo:list-item-label>
    <fo:list-item-body>
      <fo:block font-weight="normal">Marley's ghost1</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  <fo:list-item space-before.optimum="4pt">
    <fo:list-item-label>
      <fo:block margin-right="2.5pt" text-align="center">
        &amp;#2219;
      </fo:block>
    </fo:list-item-label>
    <fo:list-item-body>
      <fo:block font-weight="normal">The first of the three spirits 39</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  ...
</fo:list-block>
```

17.9 [4] Transform XML to XML-based page description language (creation)

```xml
<xsl:template match="tei:list">
  <fo:list-block margin-right="{$listRightMargin}"
    space-before="6pt" margin-left="15pt">
    <xsl:call-template name="setListIndents"/>
    <xsl:choose>
      <xsl:when test="@type='gloss'">
        <xsl:attribute name="margin-left">
          <xsl:choose>
            <xsl:when test="ancestor::tei:list">
              <xsl:value-of select="$listLeftGlossInnerIndent"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="$listLeftGlossIndent"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:attribute>
      </xsl:when>
      <xsl:otherwise>
        <xsl:attribute name="margin-left">
          <xsl:value-of select="$listLeftIndent"/>
        </xsl:attribute>
      </xsl:otherwise>
    </xsl:choose>
    <xsl:apply-templates select="tei:item"/>
  </fo:list-block>
</xsl:template>
```
18 Implementations of XSL FO

Open source

- Passive\TeX: 4000 lines of incomprehensible \TeX\ macros by an amateur, incomplete and stalled
- FoXeT: 4000 lines of \TeX\ macros by a professional, getting closer
- FOP: free-standing Java program, in the doldrums for several years

Closed source

- Antenna House: excellent full implementation, Windows only
- XEP: excellent full implementation in Java

19 FO’s good and bad points

✔ Simple to read and write, although very verbose
✔ ‘Standardised’ by a reputable body
✔ Multiple implementations
✔ Understands colour, backgrounds, fonts, URLs, Unicode etc
✗ Divorced from the typesetter
✗ Simplistic and limited page model (eg floats)
   possibly “good enough” (anathema to \TeX\-xies!)

20 Which direction?

- Forget direct \TeX interpretation of arbitrary XML…
- …embrace direct \TeX reading of constrained XML
- Forget trying to teach people \{\}…
- …embrace semantically clean markup
- Forget trying to make \TeX the centre of the universe
- …develop \TeX to keep being the best typesetting \textbf{engine}