MlBibTEX meets ConTEXt

JEAN-MICHEL HUFFLEN
LIFC (FRE CNRS 2661)
University of Franche-Comté
16, route de Gray
25030 Besançon Cedex, France
hufflen (at) lifc dot univ dash fcomte dot fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

This article reports a first experiment of using MlBibTEX — our reimplementation of BibTEX — with ConTEXt, a TEX format more modern than LATEX. We show how to take as much advantage as possible of both ConTEXt and MlBibTEX features when they are used together. Also, many end-users are accustomed to using LATEX commands inside values of BibTEX fields, and such commands may be unrecognised by ConTEXt. We explain how patterns *and* preambles *allow us to solve such problems.*

Keywords

ConTEXt, `bib` module, bibliographies, bibliography styles, BibTEX, MlBibTEX.

Introduction

Listing the bibliographical references cited within a document can be done manually — if the LATEX word processor is used, that consists of typing successive `\bibitem` commands of a `thebibliography` environment [18, § 4.3.2] — but such an approach leads to texts difficult to maintain and reuse, because they are tightly bound to *bibliography styles*. A publisher or anthology editor might like authors' last names of a 'References' section to be typeset using small capitals, whereas another publisher would require the use of standard Roman letters for these last names. Likewise, first names may be abbreviated or put *in extenso*, w.r.t. the bibliography style used. In addition, doing a document's bibliography manually is error-prone: if this bibliography is *unsorted*, that is, if the order of items is the order of first citations of these items throughout the document, some change within the document's body can cause the bibliography to be reorganised. Likewise, keys based on the author-date system [19, § 12.3] may need to be recomputed if the bibliography is enriched.

A better method is to use a *bibliography processor*: such a program is given *citation keys*, searches bibliography database files for resources associated with these keys, and arranges them according to a bibliography style, the result being a source file for a 'References' section, suitable for a word processor. A well-known association between a word and bibliography processor is given by LATEX and BibTEX [21], working in tandem, although this example is not unique. As another example, Tib [1] has sometimes been used with *Plain TEX* [17]; more generally, other examples of bibliography processors are given in [25].

'Historically', BibTEX was initially designed to work with Scribe[1] [24]. In fact, only a few points related to TEX are hard-wired within BibTEX: using braces as delimiters, considering a group such as '`{\command ...}`' as an accent command applied to arguments in order to produce a single character [19, § 13.2.2], the use of the '~' character for unbreakable spaces when names are formatted [20, § 5.4], the `width$` function, provided by the style language, that returns the width of a string, expressed using TEX units [19, Table 13.8]. Thus bibliographic entries specified with BibTEX should be usable with any *format* built from TEX, provided that end-users do not put LATEX-specific commands inside field values. Let us recall that TEX basically provides a powerful framework

[1] That is why BibTEX uses the '@' character for specifying its commands and entry types: this character introduces a command name in Scribe, like '\' in TEX. This convention is also used within Texinfo [3], the GNU documentation format.

```
@BOOK{meaney2003,
        AUTHOR = {John Meaney},
        TITLE = {Context},
        PUBLISHER = {Bantam Books},
        YEAR = 2003,
        NUMBER = 2,
        SERIES = {The \emph{Nulapeiron}
                  Sequence},
        NOTE = {The Sequel to ``Paradox''.
                [Pas de version française
                 connue!] ! french},
        LANGUAGE = english}
```

Figure 1: Example of a bibliographical entry.

to format texts nicely, but to be fit for use, the definitions of this framework need to be organised in a format. The first formats were *Plain TEX* and LATEX; another format, more modern, is ConTEXt [5]. A bibliographic module, usable in conjunction with BibTEX, has been added to ConTEXt [6, 8]. This module defines ConTEXt commands to deal with the components (metadata) of bibliographical information.[2]

Over the last few years, we have designed and implemented a 'new BibTEX'—MlBibTEX, for '**M**ulti**L**ingual BibTEX'. Of course, it has been designed to work with LATEX, but we plan to use it for other output formats, too [10]. This article is a revised and extended version associated with the presentation given at EuroTEX. It aims to report a first experiment of using MlBibTEX to build outputs suitable for ConTEXt.[3] First, we show how ConTEXt can be easily used to pretty-print bibliography database (`.bib`) files. Then we explain how an interface between ConTEXt and BibTEX can be improved when MlBibTEX is used. This article should be read without any difficulty by any user familiar with LATEX and BibTEX: it requires only basic knowledge of ConTEXt and its `bib` module. It also refers to some basic notions of XML[4] and Scheme, the implementation language for MlBibTEX.[5]

[2]If we compare this module to what is provided for LATEX, its approach is close to the `jurabib` package [19, § 12.5.1], in the sense that items of bibliographical information are given as arguments of new commands. If you would like to redefine the layout of a bibliography's items, just redefine these new commands.

[3]We have used the most recent version of ConTEXt at the time of writing, included in TEX Live 2005, available on DVD-ROM.

[4]E**X**tensible **M**arkup **L**anguage. Readers interested in an introductory book to this formalism can consult [23].

[5]The version used is described in [14].

```
<mlbiblio>
  ...
  <book id="meaney2003" language="english">
    ...
    <series>
      The <emph>Nulapeiron</emph> Sequence
    </series>
    <note>
      The Sequel to
      <emph emf="no" quotedf="yes">
        Paradox
      </emph>.
      <group language="french">
        Pas de version française connue!
      </group>
    </note>
  </book>
  ...
</mlbiblio>
```

Figure 2: XML tree for the bibliographical entry shown in Figure 1.

Pretty-print bibliographies

MlBibTEX's new syntactic features for bibliographical entries are detailed in [9]. Roughly speaking, any `.bib` file suitable for 'old' BibTEX can be processed by MlBibTEX, except that square brackets are ordinary characters for the former, syntactic delimiters for the latter. Figure 1 gives an example of a bibliographical entry for a book written in English (the value of the `LANGUAGE` field, handled by MlBibTEX). The value of the `NOTE` field includes a text to be put down only in French-speaking bibliographies, this text being enclosed by square brackets and labelled by the `french` language identifier.

As mentioned in [9], the result of parsing a `.bib` file can be viewed as an XML tree. For example, parsing a file containing the `meaney2003` entry results in the XML tree sketched in Figure 2. Such a tree can be saved into a file and displayed *verbatim* or handled by tools belonging to XML's world. ConTEXt provides a way to handle XML texts [22], so it can deal with such files. Figure 3 sketches a pretty-printer for bibliographical entries by means of ConTEXt commands documented in [7, 22], other basic TEX commands—such as `\expandafter` or `\uppercase`—being documented in [17]. These bibliographical entries are displayed using MlBibTEX's syntax. In addition, MlBibTEX's new syntax for emphasising the parts of per-

```
\enableregime[il1]

\def\ProcessMlBibTeXFieldName[#1]{{\tt \expandafter\uppercase{#1} = \textbraceleft}}
\def\CloseMlBibTeXFieldValue{{\tt \textbraceright},\par}
\def\ProcessMlBibTeXLanguagePart[#1]{\PutLanguageCommand[#1]{\tt LANGUAGE =} #1,\par}
\def\ProcessMlBibTeXNamePart[#1]{{\tt #1 =>} }
\def\PutLanguageCommand[#1]{\doifelse{#1}{english}{\language[en]}{%
  \doifelse{#1}{french}{\language[fr]}{\doif{#1}{magyar}{\language[hu]}}}}

\defineXMLenvironment[mlbiblio] \startitemize \stopitemize
\defineXMLenvironment[book] {\item {\tt @BOOK\textbraceleft}\XMLpar{book}{id}{*unkeyed*},%
 \startnarrower[left] \ProcessMlBibTeXLanguagePart[\XMLpar{book}{language}{english}]} {%
 \stopnarrower {\tt \textbraceright}}
...
\defineXMLenvironment[author] {\ProcessMlBibTeXFieldName[author]} \CloseMlBibTeXFieldValue
...
\defineXMLenvironment[first] {\ProcessMlBibTeXNamePart[first]} {, }
\defineXMLenvironment[von] {\ProcessMlBibTeXNamePart[von]} {, }
\defineXMLenvironment[last] {\ProcessMlBibTeXNamePart[last]} \unskip
\defineXMLenvironment[junior] {, \ProcessMlBibTeXNamePart[junior]} \unskip

\defineXMLenvironment[asitis] {{\tt \textbraceleft}} {{\tt \textbraceright}}
\defineXMLenvironment[emph] {\doifelse{emph}{quotedf}{yes}{{\tt ``}}{%
 \doifelse{emph}{emf}{yes}{{\tt \textbackslash emph\textbraceleft}\bgroup\em}{}}} {%
 \doifelse{emph}{emf}{yes}{{\tt \textbraceright}\egroup}{%
 \doifelse{emph}{quotedf}{yes}{{\tt ''}}{}}}

\def\GroupMarker{! }
\defineXMLenvironment[foreigngroup] {{\tt [}%
 \bgroup\PutLanguageCommand[\XMLpar{foreigngroup}{language}{*error*}]} {%
 \egroup{\tt ] : \XMLpar{foreigngroup}{language}{*error*}} }
\defineXMLenvironment[group] {\startnarrower[left]{\tt [}%
 \bgroup\PutLanguageCommand[\XMLpar{group}{language}{*error*}]} {%
 \egroup{\tt ] \GroupMarker \XMLpar{group}{language}{*error*}} \stopnarrower}
\defineXMLenvironment[nonemptyinformation] {{\tt []}\def\GroupMarker{* }} {}

\starttext
\processXMLfilegrouped{...}
\stoptext
```

Figure 3: Pretty-printing an XML tree resulting from parsing `.bib` files.

son names [21, § 4], based on keywords, is used. For example:

```
AUTHOR = {first => John, last => Meaney},
```

We can notice that keywords, syntactic delimiters, and field names are typeset using a typewriter font, whereas the Roman typeface is used for metadata. As another pretty-printing feature, typographical effects are put into action. For example, the value of the `SERIES` field will be rendered as follows:

```
SERIES = {The \emph{Nulapeiron} Sequence},
```

Likewise, any information is typeset using the typographical conventions of its own language:

```
NOTE =
{ ... [Pas de version française connue !] ... }
```

where the exclamation mark is preceded by a thin space character, as in French.

If we look at the text given in Figure 3, we notice that the only heavy part concerns *language identifiers*. ConTEXt uses ISO codes for languages [5, Ch. 7] and can switch to any language via the `\language[...]` command, without any preliminary declaration such as LaTeX needs when the `babel` package is loaded [19, § 9.2]. MlBibTeX's language identifiers are unambiguous prefixes of packages or options of the `babel` package, as explained in [12]. For example:

- `polish` is for the option of the `babel` package,
- `polski` is for the `polski` package [4, App. F],
- `pol` is for either of these last two, the final choice depends on what a user puts in the document's preamble,
- `po` is ambiguous because it may be the prefix of '*Po*lish' or '*Po*rtuguese'.

In fact, we need a complete correspondence table, with our `\PutLanguageCommand` command given in Figure 3 implementing it only partially.[6] Let us remark that such a correspondence table could be useful for other purposes, e.g., generating bibliographies for documents written using DocBook[7] [27]. This table between the structure managing MlBibTEX's language identifiers [12] and ISO codes for languages [2] has been implemented within the Scheme functions of MlBibTEX.[8]

Figure 3 gives a rough version of such a pretty-printer, which may be improved.[9] For example, the error cases are just labelled within the source text by identifiers surrounded by '`*`', which could be refined into a more efficient marking of errors. We can also align '=' signs vertically between field names and values. That can be done in a tabulate environment, but leads to slighly complicated ConTEXt commands, because we need to collect the content of a table before formatting it.

ConTEXt and MlBibTEX together

If you would like BibTEX to generate specifications of publications suitable for ConTEXt from bibliographical entries, you may use the `\setupbibtex` command, as explained in [8, § 2.4]. This command gets access to bibliography styles suitable for ConTEXt, that is, handled by the `bib` module. Since MlBibTEX can process bibliography styles using the bst language [20] in compatibility mode [13], it can deal with these styles. However, we do not recommend this solution, which should be temporary, from our point of view. In addition, the compatibility mode is not very efficient for sake of implementation issues.[10] A first improvement could be the development of new bibliography styles, using the nbst[11] language, close to XSLT[12] [26] and described in [9].

As mentioned in [19, § 13.5.2], the choices among different styles for displaying person names, work titles, ... causes a combinatorial explosion. Besides, all the functions of a bibliography style of BibTEX must be grouped into a unique file, so a rich library of bibliography styles for BibTEX should include a huge number of styles, each being monolithic. As explained in [11], several fragments of a bibliography style written using the nbst language can be assembled dynamically, provided that there is no conflict among the templates programmed using nbst. Consequently, designing styles according to a modular approach is easier in MlBibTEX than in BibTEX. Moreover, an extended version of the `\setupbibtex` command could allow the use of *several complementary files* for a bibliography style.

ConTEXt vs LATEX

End users sometimes put LATEX commands within the values of BibTEX fields. Some commands aim to increase the expressive power of the information put into `.bib` files, an example being given by the value of the `SERIES` field in Figure 1. Other examples are related to some features of BibTEX:

```
{Maria {\MakeUppercase{d}e La} Cruz}
```

—given in [19, p. 767] about person names—allows BibTEX to interpret '`{de La}`' as a particle,[13] because this group, surrounded by braces, begins with a lowercase letter, even if this particle should be typeset as 'De La'. Some commands are recognised by ConTEXt, some not. There are two solutions to this problem:

- when outputs for ConTEXt are produced, the contents of `@preamble` rubrics included in `.bib` files

[6] In fact, this `\PutLanguageCommand` command could be written in an easier way, since complete names such as `english`, `french`, ... also work as arguments of the `\language` command of ConTEXt. However, this feature is not described in [5].

[7] DocBook is an XML-based system for writing structured documents.

[8] How to put this implementation into action is shown in Figure 4.

[9] A more elaborated version can be downloaded from MlBibTEX's home page: `http://lifc.univ-fcomte.fr/~hufflen/texts/mlbibtex/contextstuff/`.

[10] When MlBibTEX parses a `.bib` file, it tries to organise information into a deep tree, as far as possible. For example, the components of a person name are split into subtrees. When the compatibility mode is used, these components are serialised into a string, and destructured again by the `format.name$` function of bst [20].

[11] **N**ew **B**ibliography **ST**yles.

[12] e**X**tensible **S**tylesheet **L**anguage **T**ransformations, the language of transformations used for XML texts.

[13] '`Maria`' being the first name, '`Cruz`' the last name.

```
(and-let* ((((log-output-p-pv 'open) jobname))                  ; Opening the log (.mblg) file.
           (((bibtexkey-alist-pv 'add-key) "hoekwater2001"))    ; Citation key to be processed.
           ...
           (((bibtexkey-alist-pv 'extend)))   ; If we want to process all the entries (\nocite{*}).
           ((let ((bib-suffix ".bib"))
              (every (lambda (filename)   ; Parsing .bib files. If the suffix is not given, the filename-plus
                                          ; function adds it.
                       (s-parse-bib-file (filename-plus filename bib-suffix #f)))
                     bibliographyfilename-list)))
           (sxml-mlbiblio-tree (s-get-sxml-mlbiblio-tree)) ; Build the SXML tree.
           (((language-trie-pv 'use-iso-code-table))) ; Using ISO codes for all the languages.
           (((preamble-pv 'set) "contextpreamble"))   ; Using @contextpreamble{...} as preambles.
           (k1 (n-assemble-nstyles stylefilename-list)) ; Styles are assembled and compiled into a
                                                        ; so-called k1 function.
           (((output-encoding-pv 'set) 'latin1)) ; Accented letters of Latin-1 allowed in the output file.
           (((bbl-output-p-pv 'open) jobname)))  ; Opening the output file.
  (k1 sxml-mlbiblio-tree)                        ; Applying the whole style to the SXML tree.
  ((bbl-output-p-pv 'close))                     ; Closing files.
  ((log-output-p-pv 'close))
  #t)                                            ; Final result.
```

Figure 4: MlBibTEX's kernel for use with ConTEXt.

are not written as BibTEX would do; instead, MlBibTEX uses `@contextpreamble` rubrics,[14] which can be used to implement some LATEX commands in ConTEXt; switching to another preamble is controlled by an option of the MlBibTEX program;[15]

- a better solution is given by *patterns*, expressed in Scheme, replacing substrings by XML-conformant strings; for example:[16]

```
(define-pattern "\\emph{#1}"
  "<emph>#1</emph>")
```

Patterns aim to process any LATEX command included in a `.bib` file, including user-defined commands, as explained in [10]. 'General' patterns are planned for the next version, only some predefined patterns are implemented now, mostly for letters accented by means of TEX commands.[17]

This solution is more general, not limited to bibliographies usable by ConTEXt. Let us assume that you have to convert a `.bib` file into HTML,[18], and consider the following title:

```
\ConTeXt, the Manual
```

Even if displaying '`\ConTeXt`' on a Web page does not cause any error, it is better to introduce this pattern:

```
(define-pattern "\\ConTeXt"
  "<symbol name='ConTeXt'/>")
```

Now you can define a way to display this symbol within a bibliography style.

Direct interface

ConTEXt does not deal with the same auxiliary files as LATEX. Moreover, it builds an `.aux` file only if the `\setupbibtex` command is activated. Let us recall that BibTEX reads only `.aux` files, never `.tex` files. However, MlBibTEX may need to parse the preamble of a source file, as explained in [12]. Concerning outputs suitable for ConTEXt, the information of interest is the encoding: can MlBibTEX put accented letters of Latin-1 directly into the resulting file? Or does it have to use TEX accent commands?

[14] This new command does not interfere with parsing `.bib` files by 'old' BibTEX, because it looks like:
`@...{`⟨*string*⟩ `(#` ⟨*string*⟩`)*}`
where '⟨*string*⟩' is surrounded by braces or double quotes. Such a command is ignored by 'old' BibTEX.

[15] ... or see how to process in Scheme in Figure 4.

[16] Let us recall that in Scheme, the '\' character is used to escape special characters in constant strings. To include it within a string, it must be itself escaped.

[17] The internal representation uses Latin-1, accented letters of this encoding being viewed as single characters.

[18] **H**yper**T**ext **M**arkup **L**anguage.

A better solution than making useless `.aux` files and the parsing of preambles of ConTEXt documents (parts between the beginning of a document and the `\starttext` command) is to build a *driver* directly written in Scheme. Of course, this task requires some knowledge of both the Scheme programming language and the broad outlines of MlBibTEX's implementation, but the result is a small-sized program, as shown in Figure 4. You can see how to add a citation key as if it were caught from an `.aux` file, and how to get all the entries of `.bib` files as the `\nocite{*}` command of LATEX would cause to. We also show how to use a preamble command — `@...{...}` — specific for ConTEXt. Other information to be supplied is:

`jobname` (string) the base name of the main input file processed by ConTEXt;

`bibliographyfilename-list` (string list) all of the `.bib` file names to be searched;

`stylefilename-list` (string list) all the fragments of a bibliography style.

The `and-let*` macro [15] causes the sequential evaluation of the clauses of its first argument to be stopped as soon as a false value (for a failure) is returned. For example, the evaluation of the whole expression given in Figure 4 stops and returns the false value if a log file (`.mblg` file) cannot be opened. Otherwise, the non-false result of a clause may become the value of a local variable. For example, the `sxml-biblio-tree` variable is given the bibliographical entries in the SXML[19] format. Then the other arguments of the `and-let*` macro are evaluated sequentially if all the clauses succeed, that is, if there is no error in parsing `.bib` files and building the bibliography style. In our case, the style — which results in a Scheme function — is applied to the bibliographical entries, and output files are closed. Finally, the true value is returned.

Going further, the `texexec` script, which launches successive run phases of ConTEXt, could be extended to launch the MlBibTEX program.

Conclusion

When we began this task, we had written only some small-sized examples using ConTEXt and emphasising its differences with LATEX. And we were afraid we would have to reprogram some important parts of MlBibTEX. To be honest, changes were needed, but not as many as we believed. Concerning the `bib` module, we learned it more quickly than we planned. The first meeting between MlBibTEX and ConTEXt has succeeded.

[19] Scheme implementation of XML. See [16] for more information.

Acknowledgements

Many thanks to Hans Hagen and Taco Hoekwater, who kindly answered my very ConTEXt-nical questions. I am also grateful to Karl Berry, who proofread this article.

References

[1] James C. Alexander: *Tib: a TEX Bibliographic Preprocessor*. Version 2.2, see `CTAN:biblios/tib/tibdoc.tex`. 1989.

[2] Harald Tveit Alvestrand: *Request for Comments: 1766. Tags for the Identification of Languages*. UNINETT, Network Working Group. March 1995. `http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1766.html`.

[3] Robert J. Chassell and Richard M. Stallman: *Texinfo. The GNU Documentation System. Version 4.8*. `http://www.gnu.org/software/texinfo`. December 2004.

[4] Antoni Diller: *LATEX wiersz po wierszu*. Wydawnictwo Helio, Gliwice. Polish translation of *LATEX Line by Line* with an additional annex by Jan Jelowicki. 2001.

[5] Hans Hagen: *ConTEXt, the Manual*. November 2001. `http://www.pragma-ade.com/general/manuals/cont-enp.pdf`.

[6] Taco Hoekwater: "The Bibliographic Module for ConTEXt". In: *EuroTEX 2001*, pp. 61–73. Kerkrade (the Netherlands). September 2001.

[7] Taco Hoekwater: *ConTEXt System Macros. Part 1: General Macros*. 2002. `http://tex.aanhet.net/context/syst-gen-doc.pdf`.

[8] Taco Hoekwater: *ConTEXt. Module Documentation*. March 2006. `http://dl.contextgarden.net/modules/t-bib/doc/context/bib/bibmod-doc.pdf`.

[9] Jean-Michel Hufflen: "MlBibTEX's Version 1.3". *TUGboat*, Vol. 24, no. 2, pp. 249–262. July 2003.

[10] Jean-Michel Hufflen: "MlBibTEX: beyond LATEX". In: Karl Berry, Baden Hughes and Steven Peter, eds., *Preprints for the 2004 Annual Meetings*, pp. 77–84. TUG, Xanthi, Greece. August 2004.

[11] Jean-Michel Hufflen: "Making MlBibTEX Fit for a Particular Language. Example of the Polish Language". *Biuletyn GUST*, Vol. 21, pp. 14–26. 2004.

[12] Jean-Michel Hufflen: *Managing Languages within MlBibTEX*. To appear. June 2005.

[13] Jean-Michel Hufflen: "BibTEX, MlBibTEX and Bibliography Styles". *Biuletyn GUST*, Vol. 23, pp. 76–80. In *BachoTEX 2006 conference*. April 2006.

[14] Richard Kelsey, William D. Clinger, Jonathan A. Rees, Harold Abelson, Norman I. Adams IV, David H. Bartley, Gary Brooks, R. Kent Dybvig, Daniel P. Friedman, Robert Halstead, Chris Hanson, Christopher T. Haynes, Eugene Edmund Kohlbecker, Jr, Donald Oxley, Kent M. Pitman, Guillermo J. Rozas, Guy Lewis Steele, Jr, Gerald Jay Sussman and Mitchell Wand: "Revised^5 Report on the Algorithmic Language Scheme". *HOSC*, Vol. 11, no. 1, pp. 7–105. August 1998.

[15] Oleg B. Kiselyov: `and-let*`*: an* `and` *with local bindings, a guarded* `let*` *special form*. March 1999. `http://srfi.schemers.org/srfi-2/`.

[16] Oleg E. Kiselyov: *XML and Scheme*. September 2005. `http://okmij.org/ftp/Scheme/xml.html`.

[17] Donald Ervin Knuth: *Computers & Typesetting. Vol. A: The TEXbook*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1984.

[18] Leslie Lamport: *LATEX. A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[19] Frank Mittelbach and Michel Goossens, with Joannes Braams, David Carlisle, Chris A. Rowley, Christine Detig and Joachim Schrod: *The LATEX Companion*. 2nd edition. Addison-Wesley Publishing Company, Reading, Massachusetts. August 2004.

[20] Oren Patashnik: *Designing BibTEX Styles*. February 1988. Part of the BibTEX distribution.

[21] Oren Patashnik: *BibTEXing*. February 1988. Part of the BibTEX distribution.

[22] PRAGMA ADE, `http://www.pragma-ade.com/general/manuals/example.pdf`: *XML in ConTEXt*. November 2001.

[23] Erik T. Ray: *Learning XML*. O'Reilly & Associates, Inc. January 2001.

[24] Brian Keith Reid: *SCRIBE Document Production System User Manual*. Technical Report, Unilogic, Ltd. 1984.

[25] David Rhead: *The "Operational Requirement" (?) for Support of Bibliographic References by LATEX 3*. Technical Report L3–005, LATEX 3. August 1993.

[26] W3C: *XSL Transformations (XSLT). Version 1.0*. W3C Recommendation. Edited by James Clark. November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`.

[27] Norman Walsh and Leonard Muellner: *DocBook: The Definitive Guide*. O'Reilly & Associates, Inc. October 1999.