A Tour around MlBibTeX and Its Implementation(s)

Jean-Michel HUFFLEN
LIFC (FRE CNRS 2661)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
FRANCE
hufflen@lifc.univ-fcomte.fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

This article describes the components of MlBibTeX, a new implementation of BibTeX including multilingual features. We justify our choices and show why our use of XML eases most operations performed by MlBibTeX. Besides, there are two implementations of MlBibTeX, a prototype developed in Scheme, and a more robust program written in C. We also explain how we take advantage of this approach.
Keywords Bibliographies, bibliography styles, BibTeX, MlBibTeX, XML processing.

Streszczenie

Ten artikuł opisuje składniki MlBibTeX-a — nowej implementacji BibTeX-a, zawierającej obsługę wielojęzyczności. Uzasadniamy nasz wybór i pokazujemy dlaczego użycie XML-a ułatwia większość operacji wykonywanych przez MlBibTeX. Istnieją dwie implementacje MlBibTeX-a — prototyp rozwijany w Scheme oraz bardziej funkcjonaly program napisany w C. Jednocześnie wyjaśniamy w jaki sposób można odnieść korzyści przy powyższym podejściu.
Słowa kluczowe Bibliographie, style bibliographiczne, BibTeX, MlBibTeX, XML.

Introduction

This article aims to give the broad outlines of the present implementation of MlBibTeX (for '**M**ultilingual BibTeX'), that we have developed since December 2002. Like its predecessor BibTeX [31], this program is a **bibliography processor** that uses *keys* cited throughout a document, searches *bibliography data bases* for these keys, and builds the 'References' section of the document. So authors do not have to type such a section themselves, they just have to ensure that the keys they use point to actual items within their bibliography data bases.

Let us give a first example of using MlBibTeX in association with the LaTeX word processor [26]: if an end-user of LaTeX cites an item by using the command `\cite{prus1897}` within a document (a .tex file), MlBibTeX can search files containing **bibliographical entries** (.bib files). These files are specified by the `\bibliography` command and such entries are expressed using this format:

```
@BOOK{pruss1897,
        AUTHOR = {Boles{\l}aw Prus},
        TITLE = {Faraon},
        PUBLISHER = {Oficyna Wydawnicza
                     GMP},
        ADDRESS = {Pozna\'{n}},
        EDITION = 1,
        NOTE = {[Title of the English
                 translation: ``The
                 Pharaoh''] ! english},
        YEAR = 1897,
        LANGUAGE = polish}
```

The result of MlBibTeX is a file containing information about the items cited throughout a document (.bbl file). When LaTeX runs again, this file is used to build the 'References' section of the document: that yields a list of **bibliographical references** like:

[1] Bolesław Prus. *Faraon.* Oficyna Wydawnicza GMP, Poznań, first edition, 1897.

Title of the English translation: "The Pharaoh".

In [11], we showed how we extended BibTeX's syntax by new features related to multilinguism:

- `[...] ! ⟨idf⟩`—'`!`' being for 'only'—is used for strings that are put when the language of the reference is *`idf`*;
- `[...] * ⟨idf⟩` is analogous, but a sequence of `*`-tags is for information that should never be empty, whatever the language of the reference is;
- `[...] : ⟨idf⟩` is a language change and is used when foreign words—w.r.t. the value of the `LANGUAGE` field—occur within a bibliographical entry.

As the previous example shows it, MlBibTeX considers the language of an entry, but can generate a reference for another language. For example, the reference given above is for a document written in English and the source processed by LaTeX is:

```
\bibitem{pruss1897}
\begin{otherlanguage*}{polish}
Boles{\l}aw Prus. \emph{Faraon}. Oficyna
Wydawnicza GMP, Pozna\'{n},
\foreignlanguage{english}{first edition},
1897. \foreignlanguage{english}{Title of
the English translation:
\begin{bblquotedtitle}The Pharaoh
\end{bblquotedtitle}}.
\end{otherlanguage*}
```

provided that the babel package [3] is loaded with the english and polish options when the whole document is processed. (Let us notice that MlBibTeX users are encouraged to use this package, but do not have to: see [18] for more details about other possible solutions.) If the reference to Entry `prus1897` is to be put at the end of a book written in Polish, 'first edition' will be replaced by '*1. wydanie*' and the value of the `NOTE` field will not appear, because it is to be put only in documents in English. When a complete 'References' section is to be generated, there are two approaches to do that: document-dependent and reference-dependent; see [13] for more details.

Concerning the layout of the references to be generated, MlBibTeX—like BibTeX—uses **bibliography styles**, a style being specified for a document by the LaTeX command `\bibliographystyle`. Here and in the bibliography of this article, we use a 'plain' style, that is, references are labelled with numbers. Other choices are possible: a 'long' style, where labels are the last name of the first author, followed by the year of the item ('[Prus 1897]' for the example above), or an 'alpha' style, where last names are abbreviated by retaining only the first three letters ('[Pru1897]'). The description of other BibTeX styles can be found in [8, § 13.2].

As we explained in [16], the use of the bst language, originating from BibTeX [30], for our bibliography styles would have led to heavy and complicated style files, hard to maintain. As a matter of fact, this language, based on handling a stack, is old and not modular. So, MlBibTeX's Version 1.3 uses a new language for bibliography styles, so-called nbst (for '**n**ew **b**ibliography **st**yle language'), close to XSLT[1] [42] and described in [16, 17]. If users developed some personal bibliography styles, they can use them within the new framework, as shown in [15].

This paper does not focus on particular technical details, but aims to describe the implementation of MlBibTeX's Version 1.3. As abovementioned, this version uses some features related to XML[2] and XSLT, so this tour around MlBibTeX's modules allows us to show that XML eases our implementation. In addition, we show how a tool, first designed to work using LaTeX-like syntax, has been enlarged in order to take advantage of XML features.

Reading this paper requires only a basic knowledge of XML, given by an introductory document such as [34]. We describe the role of each module in the first section. Then Section 'Discussion' explains our choices. Section 'Future Directions' mentions some possible evolutions for our architecture.

Architecture of our implementation

Here we describe the modules of MlBibTeX's implementation. MlBibTeX has been written using the C and Scheme programming languages—we will go thoroughly into this point in next section (Section 'Discussion')—so this notion of modules is especially important from a point of view related to conception, but not only: our programs follow strict conventions for naming types and functions, prefixed by the module's name. So we can consider that this notion of modules is directly reflected at the programming level, even if programming languages such as C and Scheme do not include this notion of modules.

Looking for files To look for the files MlBibTeX needs, it uses functions provided by the kpathsea library [35]. This library—now provided with TeX—uses databases (ls-R files) to locate files throughout a set of directories. Search rules can be extended by

[1] e**X**tensible **S**tylesheet **L**anguage **T**ransformations.
[2] e**X**tensible **M**arkup **L**anguage.

means of *environment variables*: for example, the following statement expressed using the bash[3] *shell*:

```
export MLBIBINPUTS=:/.../my-bibliography
```

—the value of this variable is a directory list whose items are separated by ':', the corresponding statement in the Windows operating system being:

```
set MLBIBINPUTS=;/.../my-bibliography
```

—tells MlBibTEX how to search for bibliography files. Other environment variables are used to search for bibliography style files, according to what is to be generated: output suitable for LATEX, of for another word processor...

Parsing bibliographical entries Within the first version of MlBibTEX [12], the scanner and parser for bibliography files were built using flex and bison, the two GNU[4] tools equivalent to lex and yacc [27], well-known since the first versions of the UNIX operating system. For sake of efficency, the scanner and parser of the present version have been rebuilt from scratch and now use an LL(1)-grammar[5]. Parsing bibliography files results in a DOM[6] tree, but entries not cited by users are skipped. So this approach can be related to a SAX[7] parser, since all the entries included in bibliographical files are not necessarily processed. In addition, the characters for the text nodes belonging to our DOM trees are encoded w.r.t. Unicode [36]. In other words, the TEX or LATEX commands used for letters with diacritical signs are replaced by the Unicode value of these letters: for example, when Entry `prus1897` is processed, the '`{\l}`' sequence (see the value associated with the `AUTHOR` field) is replaced by the Unicode value for the 'ł' Polish letter. Likewise, groups surrounded by double quotes ('`"`') are implemented by elements[8].

Dealing with languages Using multilingual features included in bibliographical entries should not cause errors when the text is input by a word processor. This is not a problem if you generate a bibliography for a document written in DocBook, an XML-based system for writing structured documents [46], or ConTEXt, a format built out of TEX[9], defined by Hans Hagen [10]: in these cases, all the languages, specified by means of a two-letter language code, optionally followed by a two-letter country code [2]—for example, '`de`', '`en-UK`', '`en-US`', '`po`'—are supposed to be known. This is not always the case: for example, the use of the babel package [3] of LATEX to write in particular languages is static: all the languages wished by an user must be selected at the beginning of processing a document. For example:

```
\usepackage[%
 french,german,polish,english]{babel}
```

means that a document whose defaut language is English (the last option of the babel package) can include parts written in French, German or Polish (in which case LATEX will be able to switch to accurate hyphenation patterns, in particular). Languages allowed for a LATEX document cannot be extended dynamically. So finding the languages whose texts can be actually input by a word processor may be a non-trivial task: we have to look into the source files expressed using LATEX's syntax[10]. Likewise, the language of a document can be difficult to determine: in fact, it defaults to English, unless a multilingual package (babel, french [7], german [33], polski [6, § F7]) is used. Another way to specify the language of a document consists of using an option of MlBibTEX:

```
mlbibtex faraon --language=polish
```

although we advise users against this feature when MlBibTEX works in association with LATEX: multilingual functions are not used, so some parts of the resulting text can be processed incorrectly.

Handling bibliography styles Here we explain how bibliography style files are organised for outputs suitable for LATEX. That is the same for outputs usable by other word processors, except that different environment variables are used (cf. Subsection 'Looking for files'). Let us recall that MlBibTEX tries to determine which languages can be used (see the previous subsection). So, let L_1, ..., L_n be these languages, If the bibliography style chosen is S, MlBibTEX may use the following files if they exist:

[3] **B**ourne **A**gain **SH**ell.That is the *shell* most commonly used with the Linux operating system. See [29] for more details.

[4] GNU's **N**ot **U**nix.

[5] Readers unfamiliar with this terminology related to compiling can refer to [1] for more details.

[6] **D**ocument **O**bject **M**odel. This is a W3C recommendation for a standard tree-based programming [34, p. 306–308], very often used to implement XML trees.

[7] **S**imple API (Application Programming Interface) for XML. This kind of parser calls functions users have to fill in (*callbacks*). So it is easier, within this approach, to retain only some particular subtrees from an XML file [34, p. 289–292]. Other applications can use a DOM parser, but this term means that the *whole* XML tree is implemented by a DOM tree.

[8] When we generate a reference for LATEX, such groups use the `bblquotedtitle` environment, as shown in the introduction. The expansion of this environment depends on the language used [18].

[9] TEX, defined by Donald E. Knuth [24], provides a general framework to format texts. To be fit for use, the definitions of this framework need to be organised in a *format*. Such a format is LATEX, others are *plain TEX* and ConTEXt.

[10] This is the case presently. But we think that this point could be improved, in connection with a new version of the babel package.We go thoroughly into that in [19].

```
<nbst:template name="format.day.month">
  <nbst:if test="month">
    <nbst:if test="day">
      <nbst:value-of select="day"/>
      <nbst:text> </nbst:text>
    </nbst:if>
    <nbst:apply-templates select="month"/>
  </nbst:if>
</nbst:template>
```

Figure 1: Putting a day number and a month name with nbst.

- *S*.nbst,
- -L_1.nbst, ..., -L_n.nbst: these bibliography style files contain *general* definitions suitable for the languages L_1, ..., L_n,
- *S*-L_1.nbst, ..., *S*-L_n.nbst: they are used to store redefinitions for the *S* bibliography style, when it is used within the languages L_1, ..., L_n.

An example using the nbst language for bibliography styles is given in Figure 1. As abovementioned, this language is close to XLST. The given template processes the day number (supposed to be the value of a so-called DAY field[11]) and the month name within a date. Using XML-based syntax, an example could be:

```
...<month><apr/></month><day>7</day>...
```

The day number may be absent, in which case, only the month name is put down. The template given in Figure 1 is fine... in English (and in French), where month names are always written the same, whether or not a day number is present:

7 April *April*

but not in Polish:

7 kwietnia *kwiecień*

because of *declensions*: if a month name follows a day number, we have to use the *genitive* case.

As explained in [16], the nbst language allows bibliography style writers to refine some operations for a particular language, an example being given in Figure 2. The language attribute of a template is inherited when other templates are invoked and a template with this attribute has higher priority than a template without it. In addition, this example shows how *modes* (like in XSLT [42, § 5.7]) can be

```
<nbst:template name="format.day.month"
               language="polish">
  <nbst:if test="month">
    <nbst:choose>
      <nbst:when test="day">
        <nbst:value-of select="day"/>
        <nbst:text> </nbst:text>
        <nbst:apply-templates
          select="month"
          mode="genitive"/>
      </nbst:when>
      <nbst:otherwise>
        <nbst:apply-templates
          select="month"/>
      </nbst:otherwise>
    </nbst:choose>
  </nbst:if>
</nbst:template>
```

Figure 2: Putting a day number and a month name in Polish with nbst.

used. The template given in Figure 2 will work with definitions for month names such as:

```
<nbst:template match="apr"
               language="polish">
  <nbst:text>kwiecień</nbst:text>
</nbst:template>
```

```
<nbst:template match="apr"
               language="polish"
               mode="genitive">
  <nbst:text>kwietnia</nbst:text>
</nbst:template>
```

As it can be seen in Figures 1 and 2, our expressions selecting parts of a bibliographical item are close to XPath's expressions [41]. We added some functions for operations difficult to perform with the functions provided by the first version of XPath[12], all these functions being documented in [17]. A few functions used within nbst texts are implemented by means of calling external functions written using the implementation's language.

Discussion

History LaTeX is widely used in the world and, as abovementioned, 'old' BibTeX is the bibliography program most commonly used with LaTeX. So BibTeX has succeeded but, as a consequence, many

[11] This field does not belong to the 'standard' fields of BibTeX. In [19], we explain why it is easy to add new fields in MlBibTeX. Let us remark that if we cite an article from a daily, this information is relevant.

[12] Dealing with string and types should be easier in XPath 2.0 [44], so a future version of MlBibTeX might be fully conformant with path expressions used in XML.

end-users have a huge number of bibliography files according to the .bib format and they wish to be still able to use them. Since the beginning of our project, we have been aware of this point: when we personally talked to some people about our reimplementation, they immediately asked us to know if their .bib files would be reusable. So even if there exists another—more user-friendly—syntax, a bibliography tool that aims to do better than BibTEX must process the .bib format.

A workaround would consist of using *converters* to XML-like syntax. such tools exist [9, 47] but, from our viewpoint, there are two drawbacks:

- such tools are designed for a transitional period of time, before XML's syntax becomes well-known: this is a defensible position within the world of computer scientists, but this transitional period might be longer for other people;
- let us agree for a direct processing of bibliography files using XML-like syntax, this processing should include a validation step. But with respect to what? Presently, we can consider that the organisation of our trees for bibliographical entries can be modelled by a DTD[13]. But this approach is poor about types. In addition, it is rigid: for example, it is difficult for the designer of a new style to add a bibliographical field, as we did with the DAY field, processed by the templates given in Figures 1 and 2. Since it should be possible to define types and constraints for the fields of bibliographical entries, we are presently working on expressing such entries with *schemas*[14]. When this work reaches maturity, we plan to allow users to specify bibliographical entries using XML-like syntax, and users will be able to validate their entries. Besides, if end-users can access the XML version of bibliographical items directly, they could also use an XQuery-like language [40] to search bibliographical data bases.

nbst vs XSLT We think that users knowing XSLT should learn nbst without difficulty, because most elements have the same behaviour, most attributes have the same meaning. Roughly speaking, the differences are:

(i) implicit rules for files associated with a style (*cf.* Subsection 'Handling bibliography styles');

(ii) the use of the `language` attribute when a template is selected;

(iii) a slightly different library of functions associated with our path expressions.

Of course, the main difference is (ii). No one can claim to know all the languages, so it is important that other people than MlBibTEX's programmers can write adaptations of styles to a particular language. Besides, such adaptation should be modular in the sense that a refinement for a language should be written separately. That would not be the case if we wrote:

```
<nbst:template name="format.day.month">
  <nbst:choose>
    <nbst:when test="language = 'polish'">
      ...
    </nbst:when>
    ...
    <nbst:otherwise>...</nbst:otherwise>
  </nbst:choose>
</nbst:template>
```

In addition, the use of modes, like in XSLT [41, § 5.7], would be tedious for implementing multilingual features because a template without mode could not be instantiated if there is no refinement for a particular language. Besides, modes can be used for declensions, as we showed it in Figure 2. In fact, nbst uses XSLT-like templates according to an object-oriented approach for languages: general methods that can be redefined. Likewise, splitting a bibliography style into several files with implicit search rules should ease the adaptation of such a style for non-computer scientists.

Other languages for bibliography styles Other projects aiming to replace BibTEX use programming languages: [25] uses Common Lisp, BibTEX++ [21] uses Java, Bibulus [47] uses Perl. We think that these languages are better than bst, the language used by 'old' BibTEX... but from the viewpoint of a computer scientist. Editors of proceedings or book series, that is, litterary people, may have to write own bibliography styles. Even if there is lack of background about using XSLT, we think that a language that allows pattern-matching should be better for non-computer scientists.

Language(s) for implementation(s) In [12], we explained why we programmed MlBibTEX in C: for sake of efficiency and portability. The implementation of MlBibTEX's Version 1.3 uses functions of the

[13] **D**ocument **T**ype **D**efinition. A DTD defines a document markup model [34, Ch.5].

[14] Schemas have more expressive power than DTDs, in the sense that they are more modular, they allow users to define types precisely, which makes more precise the validation of a XML text with respect to a schema. In addition, this approach is more homogeneous since schemas are XML texts, whereas DTDs are not. There are four ways to specify schemas: Relax NG [5], Schematron [20], Examplotron [39], XML Schema [45].

```
<nbst:template match="foreigngroup">
  <nbst:value-of select="call(language_open_change,@language)"/>
  <!--  If the babel package is used and an accurate option has been selected, this external function puts
        the \foreignlanguage command...
    -->
  <nbst:apply-templates use-language="@language"/>
  <nbst:value-of select="call(language_close_change,@language)"/>
  <!--  ... and this external one puts a closing brace.      -->
</nbst:template>
```

Figure 3: Example of calling an external function.

libxml2 [37] and gdome2[15] [4] libraries, both written using C. In particular, we directly use the functions related to the DOM trees, and the DOM parser of libxml2 for nbst files.

Programming using C being a long task, we have also developed prototypes using Scheme. The first prototype was an interpreter of the bst language of BibTeX [14]. We are still using it, it was useful to implement the joint use of nbst and bst, described in [15]. The second was MlBibTeX's Version 1.2, when we experienced an extended syntax for person names [16]. And the first working version of the present version (1.3) has been written using Scheme, too. The features related to XML have been implemented by means of SSAX[16] [22], and SXPath[17] [23], a SAX parser and an implementation of XPath using SXML[18], an implementation of XML trees by means of Scheme expressions. The only drawback of this prototype: we cannot mix functions written using C and Scheme, even if an interface between these two languages is available, because SXML is comparable with DOM, but is not DOM. The only non-portable point of this implementation is the search of files (cf. Subsection 'Looking for files'), which uses non-standard functions for calling commands of the kpathsea library.

Even if these Scheme are not as efficient as a C program, they allowed us to get much experience with some real-sized cases. We began to write bibliography styles using XSLT, in order to study the expressive power of this language. Then we were able to develop an nbst processor quickly, so we were able to appreciate how accurate our choices about nbst were.

On another point, some people can argue that C is not a high-level programming language in comparison with other languages like Scheme, CAML[19], Common Lisp, Java, Perl[20], Python, Standard ML[21], Ruby, or others... but we noticed that the functions of libxml2 and gdome2 are more efficient than the corresponding tools written using Java, often used to deal with XML texts. We wrote our own implementation of nbst using C, after having got much experience by studying the implementation of XSLT given by the libxslt library [38], written using C and built out of libxml2.

From our personal viewpoint, the only trouble caused by programming in C is about the external functions called when nbst templates are instantiated. (Of course, the external functions we call are written using Scheme when we use our version in Scheme.) An example of using such functions is given in Figure 3: a template that processes language changes, expressed by '`[...]  :` ⟨*idf*⟩' within a bibliographical entry, and implemented as:

```
<foreigngroup language=⟨idf⟩>
  ...
</foreigngroup>
```

in our representation in XML. Let us remark that all the operations performed this way are low-level: for example, determining which languages can be processed when the babel package is used with LaTeX (*cf.* Subsection 'Dealing with languages'). All the operations that *actually* rule the layout of a bibliography can be easily expressed using nbst. And let us remark that even if XSLT is as powerful as the Turing machine [28], its conceptors of XSLT provide a possible use of external functions[22]. So our use of external functions does not seem to us to be heretical, especially if we have to deal with a formalism other than XML, that is, LaTeX.

[15] GNOME (GNU Network Object Model Environment) DOM Engine.
[16] Scheme implementation of SAX.
[17] Scheme implementation of XPath.
[18] Scheme implementation of XML.
[19] **C**ategorical **A**bstract **M**achine **L**anguage.
[20] **P**ractical **E**xtraction **R**eport **L**anguage.
[21] **M**etalanguage.
[22] By means of the `extension-result-prefixes` attribute of the root element `xsl:stylesheet`.

Future Directions

Even if MlBibTeX was originally designed to work with LaTeX, introducing some XML processing inside it makes easier the generation of references according to other markup languages, some examples being XHTML[23], DocBook, XSL-FO[24] [43], ... This direction has reached some results, and we plan to go on. More generally, we think that MlBibTeX can become a central platform for communications between bibliographical databases and bibliographies for documents. A second step will be done when the specification of bibliographical entries using XML-like syntax is possible: we go thoroughly into these points in [19].

Conclusion

When we programmed MlBibTeX's first version, we were obviously interested in putting into action such a program. And when we demonstrated it, we realised that many people were interested in such a multilingual tool for bibliographies. More precisely, we became aware that many people would adapt it to particular languages. That is why we were careful about the definition of precise architecture. Until now, our approach seems to us to be robust, and ready to be publicly used.

Acknowledgements

Many thanks to Paweł D. Mogielnicki, who has written the Polish translation of the abstract.

References

[1] Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman: *Compilers, Principles, Techniques and Tools*. Addison-Wesley Publishing Company. 1986.

[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] Joannes Braams: *Babel, a Multilingual Package for Use with LaTeX's Standard Document Classes*. Version 3.7. May 2002. `CTAN:macros/latex/required/babel/babel.dvi`.

[23] e**X**tensible **H**yper**T**ext **M**arkup **L**anguage: that is a new version of HTML using XML-like syntax.

[24] e**X**tensible **S**tylesheet **L**anguage—**F**ormatting **O**bjects: this language aims to describe high-quality print outputs. Such documents can be processed by the `\xmltex` command (resp. the `\pdfxmltex` command), from PassiveTeX [32, p. 180] to get .dvi files (resp. .pdf files).

[4] Paolo Casarini and Luca Padovani: "The GNOME DOM Engine". In: *Extreme Markup Languages 2001 Conference*. March 2001.

[5] James Clark *et al.*: *Relax NG*. `http://www.oasis-open.org/committees/relax-ng/`. 2002.

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

[7] Bernard Gaulle : *Notice d'utilisation du style french multilingue pour LaTeX*. Version pro V5.01. Janvier 2001. `CTAN:loria/language/french/pro/french/ALIRE.pdf`.

[8] Michel Goossens, Frank Mittelbach and Alexander Samarin: *The LaTeX Companion*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1994.

[9] Vidar Bronken Gundersen and Zeger W. Hendrikse: *BibTeX as XML Markup*. January 2003. `http://bibtexml.sourceforge.net`.

[10] Hans Hagen: *ConTeXt, the Manual*. November 2001. `http://www.pragma-ade.com`.

[11] Jean-Michel Hufflen: "MlBibTeX: a New Implementation of BibTeX". In: *EuroTeX 2001*, (p. 74–94). Kerkrade, The Netherlands. September 2001.

[12] Jean-Michel Hufflen: "Lessons from a Bibliography Program's Reimplementation". In: LDTA *2002*, Vol. 65.3 of *Electronic Notes in Theoretical Computer Science*. Elsevier, Grenoble, France. April 2002.

[13] Jean-Michel Hufflen: "Multilingual Features for Bibliography Programs: from XML to MlBibTeX". In: *EuroTeX 2002*, (p. 46–59). Bachotek, Poland. April 2002.

[14] Jean-Michel Hufflen: *Interaktive BibTeX-Programmierung*. DANTE, Herbsttagung 2002, Augsburg. Oktober 2002.

[15] Jean-Michel Hufflen: "Mixing Two Bibliography Style Languages". In: LDTA *2003*, Vol. 82.3 of *Electronic Notes in Theoretical Computer Science*. Elsevier, Warsaw, Poland. April 2003.

[16] Jean-Michel Hufflen: "European Bibliography Styles and MlBibTeX". *TUGboat*, Vol. 24, no. 1. EuroTeX 2003, Brest, France. June 2003.

[17] Jean-Michel Hufflen: *MlBibTeX's Version 1.3*. TUG 2003, Outrigger Waikoloa Beach Resort, Hawaii. July 2003.

[18] Jean-Michel HUFFLEN: "Making MlBibTEX Fit for a Particular Language. Example of the Polish Language". *Biuletyn* GUST. In print. Presented at the BachoTEX 2003 conference. 2004.

[19] Jean-Michel HUFFLEN: *MlBibTEX beyond LATEX*. TUG 2004, Xanthi, Greece. August 2004.

[20] ISO/IEC 19757: *The Schematron. An* XML *Structure Validation Language Using Patterns in Trees*. `http://www.ascc.net/xml/resource/schematron/schematron.html`. June 2003.

[21] Ronan KERYELL: "BibTEX++: Towards Higher-Order BibTEXing". In: *EuroTEX 2003*, (p. 143). ENSTB. June 2003.

[22] Oleg KISELYOV: "A Better XML Parser through Functional Programming". In: *4th International Symposium on Practical Aspects of Declarative Languages*, Vol. 2257 of *Lecture Notes in Computer Science*. Springer-Verlag. 2002.

[23] Oleg KISELYOV and Kirill LISOVSKY: "XML, XPath, XSLT Implementations as SXML, SXPath, and SXSLT". In: *International Lisp Conference 2002*. October 2002.

[24] Donald Ervin KNUTH: *Computers & Typesetting. Vol. A: the TEXbook*. Addison-Wesley Publishing Company, Reading, Massachusetts. 1984.

[25] Matthias KÖPPE: *A BibTEX System in Common Lisp*. January 2003. `http://www.nongnu.org/cl-bibtex`.

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

[27] John LEVINE, Tony MASON and Doug BROWN: *lex & yacc*. 2nd edition. O'Reilly & Associates, Inc. October 1992.

[28] Bob LYONS: *Universal Turing Machine in* XSLT. `http://www.unidex.com/turing/utm.htm`. March 2001.

[29] Cameron NEWHAM and Bill ROSENBLATT: *Learning the bash Shell*. 2nd edition. O'Reilly & Associates, Inc. January 1998.

[30] Oren PATASHNIK: *Designing BibTEX styles*. February 1988. Part of LATEX' distribution.

[31] Oren PATASHNIK: *BibTEXing*. February 1988. Part of LATEX' distribution.

[32] Dave PAWSON: XSL-FO. O'Reilly & Associates, Inc. August 2002.

[33] Bernd RAICHLE: *Die Makropakete „german" und „ngerman" für LATEX 2ε, LATEX 2.09, Plain-TEX and andere darauf Basierende Formate*. Version 2.5. Juli 1998. Im Software LATEX.

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

[35] TUG Working Group on a TEX Directory Structure: *A Directory Structure for TEX Files. Version 0.9995*. `CTAN:tex/archive/tds/standard/tds-0.9995/tds.dvi`. January 1998.

[36] *The Unicode Standard Version 3.0*. Addison-Wesley. February 2000.

[37] Daniel VEILLARD: *The* XML *C Parser and Toolkit of Gnome. libxml*. `http://xmlsoft.org`. March 2003.

[38] Daniel VEILLARD: *The* XML *C Parser and Toolkit of Gnome*. XSLT. `http://xmlsoft.org/XSLT`. March 2003.

[39] Eric VAN DER VLIST: *Examplotron*. `http://examplotron.org`. February 2003.

[40] W3C: *XQuery 1.0: an* XML *Query Language*.

[41] W3C: XML *Path Language (XPath). Version 1.0*. W3C© Recommendation. Edited by J. Clark and S. DeRose. November 1999. `http://www.w3.org/TR/1999/REC-xpath-19991116`.

[42] W3C: XSL *Transformations (*XSLT*). Version 1.0*. W3C© Recommendation. Written by S. Adler, A. Berglund, J. Caruso, S. Deach, T. Graham, P. Grosso, E. Gutentag, A. Milowski, S. Parnell, J. Richman and S. Zilles. November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`.

[43] W3C: *Extensible Stylesheet Language (*XSL*). Version 1.0*. W3C© Recommendation. Edited by J. Clark. October 2001. `http://www.w3.org/TR/2001/REC-xsl-20011015/`.

[44] W3C: XML *Path Language (XPath) 2.0*. W3C© Working Draft. Edited by A. Berglund, S. Boag, D. Chamberlin, M. F. Fernandez, M. Kay, J. Robie and J. Siméon. November 2002. `http://www.w3.org/TR/2002/WD-xpath20-20021115`.

[45] W3C: XML *Schema*. November 2003. `http://www.w3.org/XML/Schema`.

[46] Norman WALSH and Leonard MUELLNER: *DocBook: the Definitive Guide*. O'Reilly & Associates, Inc. October 1999.

[47] Thomas WIDMAN: "Bibulus—a Perl XML Replacement for BibTEX". In: *EuroTEX 2003*, (p. 137–141). ENSTB. June 2003.