MlBibTeX: Reporting the experience*

Jean-Michel Hufflen
LIFC (EA CNRS 4157)
University of Franche-Comté
16, route de Gray
25030 BESANÇON CEDEX
France
hufflen (at) lifc dot univ-fcomte dot fr
http://lifc.univ-fcomte.fr/~hufflen

Abstract

This article reports how the different steps of the MlBibTeX project were conducted until the first public release. We particularly focus on the problems raised by reimplementing a program (BibTeX) that came out in the 1980's. Since that time, implementation techniques have evolved and new requirements have appeared, as well as new programs within TeX's galaxy. Our choices are explained and discussed.

Keywords TeX, LaTeX, BibTeX, reimplementation, reverse engineering, implementation language, program update.

Streszczenie

Artykuł omawia realizację poszczególnych kroków przedsięwzięcia MlBibTeX, w czasie do przedstawienia pierwszej publicznej wersji. W szczególności skupiamy się na problemach powstałych przy reimplementacji programu (BibTeX), powstałego w latach 80 zeszłego wieku. Od tego czasu rozwinęły się techniki implementacyjne, powstały nowe wymagania oraz nowe programy w świecie TeX-owym. Przedstawiamy i dyskutujemy dokonane wybory.

Słowa kluczowe TeX, LaTeX, BibTeX, reimplementacja, *reverse engineering*, język implementacji, aktualizacja programu.

0 Introduction

In 2003, TeX's 25th anniversary was celebrated at the TUG[1] conference, held in Hawaii [1]. LaTeX [28] and BibTeX [35] — the bibliography processor usually associated with the LaTeX word processor — are more recent, since they came out in the 1980's, shortly after TeX. All are still widely used, such longevity being exceptional for software. However, these programs are aging. Of course, recent versions have incorporated many features absent from the first versions, which proves the robustness of these systems. Nevertheless, they present some limitations due to the original conception, and a major reimplementation may be needed to integrate some modern requirements. In addition, interactive word processors have made important progress and are serious rivals, even if they do not yield typesetting of such professional quality. That is why some projects aim to provide new programs, based on TeX & Co.'s ideas.[2] A first representative example is the LaTeX 3 project [32], a second is $\mathcal{NTS}$ [27].

MlBibTeX — for '**M**ulti**L**ingual BibTeX' — belongs to the class of such projects. Let us recall that this program aims to be a 'better BibTeX', especially regarding multilingual features. For an end-user, MlBibTeX behaves exactly like 'classical' BibTeX: it searches bibliography data base (.bib) files for citation keys used in a document and then arranges the references found, writing them to a .bbl file suitable for LaTeX, w.r.t. a bibliography style. MlBibTeX is written in Scheme,[3] it uses XML[4] as a

* Title in Polish: *MlBibTeX: raport z doświadczeń*.

[1] TeX **U**sers **G**roup.

[2] Concerning TeX, an additional point is that TeX's development has been frozen by its author, Donald E. Knuth [26]. If incorporating new ideas to a 'new TeX' leads to a major reimplementation, the resulting program must be named differently.

[3] The version used is described in [24].

[4] E**X**tensible **M**arkup **L**anguage. Readers interested in an introductory book to this formalism can consult [37].

central format: when entries of .bib files are parsed, they result in an XML tree. Bibliography styles taking advantage as far as possible of MlBibTeX's new features are written using nbst,[5] a variant of XSLT[6] described in [15]. The stack-based bst language [34] used for writing bibliography styles of BibTeX can be used in a compatibility mode [20].

We think that the experience we have gained in developing MlBibTeX may be useful for other, analogous, projects. To begin, we briefly review the chronology of this development. As will be seen, this development has not been linear, and the two following sections focus on the problems we had to face. We explain how we have determined which criteria are accurate when a programming language is to be chosen for such an application. Then we show how the compatibility with 'old' data and the integration of modern features should be managed.

1 MlBibTeX's chronology

Oct. 2000 MlBibTeX's design begins: the syntax of .bib files is enriched with multilingual annotations. Version 1.1's prototype is written using the C programming language and tries to reuse parts of 'old' BibTeX's program as far as possible.

May 2001 The first article about MlBibTeX is [9]. Later, the experience of developing MlBibTeX's Version 1.1 is described in [10].

May 2002 After discussions with participants at the EuroBachoTeX conference, we realise that the conventions for bibliography styles are too diverse, even if we consider only those of European countries. We realise that this first approach is quite unsuitable, without defining a new version of the bst language. So we decide to explore two directions. First, we develop a questionnaire about problems and conventions concerning bibliography styles used within European countries. Second, we begin a prototype in Scheme implementing the bst language [11]. Initially, this prototype is devoted to experiments about improving bst in a second version, 1.2.

Jan. 2003 Version 1.2 is stalled. The new version (1.3) is based on XML formats. The nbst language is designed and presented at [12, 13]. We explain in [14] how the results of our questionnaire have influenced this new direction.

Feb. 2004 It appears to us that MlBibTeX should be developed using a very high-level programming language, higher than C. So we consider again the prototype in Scheme that we sketched in 2002. SXML[7] [25] is chosen as the representation of XML texts in Scheme. Some parts of MlBibTeX are directly reprogrammed from C to Scheme. As for the other parts, this prototype is a good basis for much experiment [16].

Nov. 2004 The version written in C is definitely dropped, whereas the version in Scheme is modified to improve efficiency; it becomes the 'official' MlBibTeX [18].

Sep. 2005 We decided to freeze MlBibTeX's design and concentrate only on finishing programming. Many Scheme functions are rewritten in conformity to SRFIs[8] [39].

May 2006 A working version is almost finished, except for the interface with the kpathsea library.

May 2007 Public availability of MlBibTeX's Version 1.3.

Let us also explain that MlBibTeX is not our only task. As an Assistant Professor in our university, we teach computer science, and participate in other projects. As a consequence, MlBibTeX's development has been somewhat anarchic: we hardly worked on it for two or three months, put it aside for one or two months, and so on. Last, we have supervised some student projects regarding graphical tools around MlBibTeX [2, 8], programmed using Ruby [31], but concerning the development of the MlBibTeX program itself, we have done it alone.

2 Choice of an implementation language

There are several programming paradigms: imperative, functional, and logic programming. There are also several ways to implement a programming language: interpretation and compilation. Some paradigms are more appropriate, according to the domain of interest. Likewise, some interpreted languages are more appropriate if you want to program a prototype quickly and are just interested in performing some experiment.[9] But compiled languages are often preferable if a program's efficiency is crucial. In addition, the level of a programming language has some influence on development: in a high-level language, low-level details of structures' implementation do not have to be made explicit, so

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

[6] e**X**tensible **S**tylesheet **L**anguage **T**ransformations, the language of transformations used for XML documents [44].

[7] Scheme implementation of XML.

[8] Scheme **R**equests for **I**mplementation, an effort to coordinate libraries and other additions to the Scheme language between implementations.

[9] Such is the case for the two graphical tools around MlBibTeX programmed in Ruby by our students [2, 8].

development is quicker, and the resulting programs are more concise, nearer to a mathematical model.

In addition to these general considerations, let us recall that we aim to replace an existing program by a new one. This new program is supposed to do better than the 'old' one. 'To do better' may mean 'to have more functionalities, more expressive power', but for sake of credibility, it is desirable for the new program to be as efficient as the 'old' one. Let us not forget that TeX and BibTeX are written using an old style of programming — more precisely, a monolithic style used in the 1970's–1980's — based mainly on global variables, without abstract data types. Choosing a language implemented efficiently is crucial: as a counter-example, $\mathcal{NTS}$, written using Java, has been reported 100 times slower than TeX [42, § 5]. That is why we wrote MlBibTeX's first version using C, because of its efficiency. In addition, this language is portable to most operating systems. And to make our program modular, we defined precise rules for naming procedures [10, § 3]. But two problems appeared.

First, MlBibTeX's development has not been a daily task, as mentioned above. Even if we are personally able to program large applications in C, it is difficult to put aside a C program and resume it later: from this point of view, C is not a very high-level language. Besides, let us not forget that we are working within an open domain, as natural languages are. A change may be needed because of new features concerning languages that had not yet been integrated into MlBibTeX's framework. The higher the level, the more easily such a change can be applied.

Second, we want end-users of MlBibTeX to be able to influence the behaviour of this program. For example, many BibTeX users put LaTeX commands inside values associated with fields of .bib files, in order to increase their expressive power within bibliographical data. These users should be able to specify how to handle such commands when .bib files are converted into XML trees. In particular, this is useful if MlBibTeX is used to produce outputs for word processors other than LaTeX [21]. How to do that in C, without defining a mini-language to express such functions? In this case, using a script language is a better choice ... provided that this language is efficient. Another choice is a Lisp[10] dialect, as in Emacs [40]: end-users can customise Emacs' behaviour by writing expressions using the Emacs Lisp language [30]. This choice is homogeneous: the entire Emacs editor is expressed in Emacs Lisp, excepting for the implementation of low-level functionalities.

Finally, our choice was Scheme, the modern dialect of Lisp. We confess that we are personally attracted by functional programming languages, because they can abstract procedures as well as data: in this sense, they are very high-level programming languages. Concerning Scheme, it seems to us to be undebatable that it has very good expressive power, and takes as much advantage as possible of lexical scoping. In addition, it allows some operations to be programmed 'impurely', by side effects, as in imperative programming, in order to increase efficiency. However, we use this feature parsimoniously, on local variables, since it breaks the principles of functional programming. We have defined precise rules for naming variables, as we did in C for the first version, in order to emphasise the modular decomposition of our program [19]. Last but not least, Scheme programs may be interpreted — when software is being developed — or compiled, in which case they are more efficient. As an example of a good Scheme implementation, bigloo [38] compiles Scheme functions by transforming them into C functions, then these C functions are compiled, in turn.

If we compare the implementations in C and Scheme, the latter is better, as expected from a very high-level programming language. But programming an application related to TeX using a language other than C reveals a drawback: the kpathsea library [3] is written in C. Let us recall that kpathsea implements functions navigating through the TDS[11] [43]. In particular, such functions localise the files containing the specification of a class for a LaTeX document or a bibliography style when BibTeX runs. If there is a compatibility mode, for 'old' bibliography styles written in bst, the functions of this compatibility mode should be able to localise such files too. Likewise, 'new' bibliography styles written in nbst, should be localised by means of an analogous method. This implies that the language — or, at least, an implementation of the language — used for our software includes an interface with C.

Of course, what we expose above proceeds from general considerations. After all, we do not know if BibTeX++ [4] — a successor of BibTeX based on Java, with bibliography styles also written in Java — is much less efficient than BibTeX. This may not be the case. The advantages of script languages in such development appear if we consider Bibulus [46], another successor of BibTeX, written using Perl.[12] It

[10] **LIS**t **P**rocessor.

[11] TeX **D**irectory **S**tructure.

[12] **P**ractical **E**xtraction **R**eport **L**anguage. A didactic introduction to this language is [45].

has developed more quickly than MlBibTeX, but is 'less' multilingual and uses BibTeX when it runs. That is, Bibulus does not replace BibTeX wholly, as MlBibTeX attempts to do. In addition, there is an example where the need of a programming language at a higher level than C appeared: the project of moving Ω — a successor of TeX — into a C++ platform [36].

We personally think that an implementation of $\mathcal{NTS}$ in Common Lisp [41] — what was planned initially — would have been preferable. As mentioned in [47], the object-oriented features of Common Lisp (CLOS[13]) have been added to the language's basis — as C++ object-oriented functionalities are added on top of C — but the language itself is not actually object-oriented. In [47], this point is viewed as a drawback. First, we personally think that not everything is an object, from the point of view of conceiving ideas. Second, Common Lisp, even if it is a functional programming language, allows some operations to be performed more efficiently by means of side effects, like Scheme.[14] But Common Lisp's standard does not specify an interface with C, as Scheme does, although some implementations provide this service. However, we personally prefer Scheme: it is simpler and more modern.

3 Choice of strategy

3.1 Languages

TeX & Co. have been wonderful programs since the date they came out. Although they behave very nicely, the syntaxes are quite archaic. TeX's is not homogeneous — although LaTeX 2ε and LaTeX 3 [32] try to correct this point — for example, different delimiters are used to change size ('`{\small ...}`') or face ('`\textbf{...}`'). BibTeX's syntax suffers from lack of expressive power: for example, the only way to put a brace within a field's value is to give its code number by '`\symbol{...}`'. 'Semantically', TeX's language provides many intelligent features, as mentioned in [6], but does not meet a modern style of programming. Likewise, .bib files' syntax can express only 'verbatim' values, except for some 'tricks' like inserting '-' characters for a range of page numbers. The specification of structured values like person or organisation names is easy for simple cases, but quickly becomes obscure in more complicated cases [22].

In addition, new syntactic sugar may be needed to meet some new requirements. As an example, [23] points out that the arguments of some macros — e.g., `\catcode` — are not easily parseable. As another example, the ConTeXt format [7] implements a homogeneous expression of setup commands, by means of a '*`key=value`*' syntax:

```
\setuplayout[backspace=4cm,topspace=2.5cm]
```

Nevertheless, is it reasonable to add more and more syntactic sugar to such old-fashioned syntax? Would the definition of new languages not be preferable? Of course, the present languages of TeX and BibTeX will still remain to be used, due to the huge number of files using them and developed by end-users. But if a new language is designed, it should become the usual way to deal with the new program. Of course, end-users will have to get used to the new language. But that can be done progressively and synergy between developers and users may cause this new language to be improved if need be.

In addition, let us remark that in our case, the new language for bibliography styles (nbst) is close to XSLT, so we think that users familiar with the former can get used to the latter easily.

3.2 New services

Now it is admitted that composite tasks are not to be done by a monolithic program, but by means of a cooperation among several programs. From this point of view, the cooperation between LaTeX and BibTeX is exemplary. But BibTeX is too strongly related to LaTeX. BibTeX can be used to build bibliographies for ConTeXt documents, but only because this word processor belongs to the TeX family. On the contrary, writing a converter from BibTeX to HTML[15] by means of the bst language is impossible without loss of quality: for example, the unbreakable space character is represented by '~' — as in TeX — when names are formatted [22], and this convention cannot be changed.[16] We see that such problems can be avoided by considering an XML-like language as a central format. In our case, generating bibliographies according to formats other than LaTeX's should be easy since the LaTeX commands end users put into .bib files are removed when these files are parsed. This point is detailed in [17, 21].

4 Conclusion

Last but not least, we have enjoyed designing and implementing MlBibTeX, even if this development backtracked several times. In addition, we think

[13] **C**ommon **L**isp **O**bject **S**ystem.

[14] Emacs Lisp, too, and the components of the Emacs editor largely use this feature.

[15] **H**yper**T**ext **M**arkup **L**anguage. Readers interested in an introduction to this language can refer to [33].

[16] In fact, there are such converters, an example being BibTeX2HTML [5], written using Objective Caml [29], a functional programming language.

that this development shows the difficulties related to such a task. Two parts have to be managed in parallel. The first part is *reverse engineering*, that is, guessing the concept from the program. The second: enlarging what already exists. In comparison with 'classical' development of a new program from scratch, tests concerning the compatibility mode are easy to perform: we can simply compare what is given by the two programs, the 'old' one becoming an oracle. But reaching a homogeneous concept is not obvious if we want to keep backward compatibility. Nevertheless, we hope that we have done some satisfactory work.

5 Acknowledgements

Many thanks to Jerzy B. Ludwichowski, who has written the Polish translation of the abstract, and to the proofreaders: Karl Berry, Barbara Beeton.

References

[1] William Adams, ed.: *TUG 2003 Proceedings*, *TUGboat*, Vol. 24:1. July 2003.

[2] Cédric Bassetti and Christian Bon: *Interactive Specification of bibliography styles for MlBibTeX*. Report of student project. University of Franche-Comté. May 2006.

[3] Karl Berry and Olaf Weber: *Kpathsea library*. `http://tug.org/kpathsea/`.

[4] Emmanuel Donin de Rosière: *From Stack Removing in Stack-Based Languages to BibTeX++*. Master's thesis, ENSTB, Brest. 2003.

[5] Jean-Christophe Filliâtre and Claude Marché: *The BibTeX2HTML Home Page*. June 2006. `http://www.lri.fr/~filliatr/bibtex2html/`.

[6] Jonathan Fine: "TeX as a Callable Function". In: *EuroTeX 2002*, pp. 26–30. Bachotek, Poland. April 2002.

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

[8] Stéphane Henry and Jérôme Voinot: *Interface for MlBibTeX. Getting Bibliographical Entries Interactively*. Report of student project. University of Franche-Comté. May 2005.

[9] Jean-Michel Hufflen : « Vers une extension multilingue de BibTeX ». *Cahiers GUTenberg*, Vol. 39–40, p. 23–38. In *Actes du Congrès GUTenberg 2001*, Metz. Mai 2001.

[10] Jean-Michel Hufflen: "Lessons from a Bibliography Program's Reimplementation". In: Mark van den Brand and Ralf Lämmel, eds., *LDTA 2002*, Vol. 65.3 of *ENTCS*. Elsevier, Grenoble, France. April 2002.

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

[12] Jean-Michel Hufflen: *Die neue Sprache für MlBibTeX*. DANTE 2003, Bremen. April 2003.

[13] Jean-Michel Hufflen: "Mixing Two Bibliography Style Languages". In: Barrett R. Bryant and João Saraiva, eds., *LDTA 2003*, Vol. 82.3 of *ENTCS*. Elsevier, Warsaw, Poland. April 2003.

[14] Jean-Michel Hufflen: "European Bibliography Styles and MlBibTeX". *TUGboat*, Vol. 24, no. 3, pp. 489–498. EuroTeX 2003, Brest, France. June 2003.

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

[16] Jean-Michel Hufflen: "A Tour around MlBibTeX and Its Implementation(s)". *Biuletyn GUST*, Vol. 20, pp. 21–28. In *BachoTeX 2004 conference*. April 2004.

[17] Jean-Michel Hufflen: "MlBibTeX: Beyond LaTeX". In: Apostolos Syropoulos, Karl Berry, Yannis Haralambous, Baden Hughes, Steven Peter and John Plaice, eds., *International Conference on TeX, XML, and Digital Typography*, Vol. 3130 of *LNCS*, pp. 203–215. Springer, Xanthi, Greece. August 2004.

[18] Jean-Michel Hufflen: *Beschreibung der MlBibTeX-Implementierung mit Scheme*. DANTE 2004, Herbsttagung, Hannover. Oktober 2004.

[19] Jean-Michel Hufflen: "Implementing a Bibliography Processor in Scheme". In: J. Michael Ashley and Michel Sperber, eds., *Proc. of the 6th Workshop on Scheme and Functional Programming*, Vol. 619 of *Indiana University Computer Science Department*, pp. 77–87. Tallinn. September 2005.

[20] Jean-Michel Hufflen: "BibTeX, MlBibTeX and Bibliography Styles". *Biuletyn GUST*, Vol. 23, pp. 76–80. In *BachoTeX 2006 conference*. April 2006.

[21] Jean-Michel Hufflen: "MlBibTeX Meets ConTeXt". *TUGboat*, Vol. 27, no. 1,

pp. 76–82. EuroTEX 2006 proceedings, Debrecen, Hungary. July 2006.

[22] Jean-Michel Hufflen: "Names in BibTEX and MlBibTEX". *TUGboat*, Vol. 27, no. 2, pp. 243–253. TUG 2006 proceedings, Marrakesh, Morocco. November 2006.

[23] David Kastrup: "Designing an Implementation Language for a TEX Successor". In: *Proc. EuroTEX 2005*, pp. 71–75. February 2005.

[24] 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.

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

[26] Donald Ervin Knuth: "The Future of TEX and METAFONT". *TUGboat*, Vol. 11, no. 4, pp. 489. December 1990.

[27] Joachim Lammarsch: "The History of $\mathcal{NTS}$". In: *EuroTEX 1999*, pp. 228–232. Heidelberg (Germany). September 1999.

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

[29] Xavier Leroy, Damien Doligez, Jacques Garrigue, Didier Rémy and Jéróme Vouillon: *The Objective Caml System. Release 0.9. Documentation and User's Manual*. 2004. `http://caml.inria.fr/pub/docs/manual-ocaml/index.html`.

[30] Bill Lewis, Dan LaLiberte, Richard M. Stallman and the GNU Manual Group: *GNU Emacs Lisp Reference Manual*. `http://www.gnu.org/software/emacs/elisp-manual/`.

[31] Yukihiro Matsumoto: *Ruby in a Nutshell*. O'Reilly. English translation by David L. Reynolds, Jr. November 2001.

[32] Frank Mittelbach and Rainer Schöpf: "Towards LATEX 3.0". *TUGboat*, Vol. 12, no. 1, pp. 74–79. March 1991.

[33] Chuck Musciano and Bill Kennedy: *HTML & XHTML: The Definitive Guide*. 5th edition. O'Reilly & Associates, Inc. August 2002.

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

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

[36] John Plaice and Paul Swoboda: "Moving Omega to a C++-Based Platform". *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, Vol. 20, pp. 3–5. In *BachoTEX 2004 conference*. April 2004.

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

[38] Manuel Serrano: *Bigloo. A Practical Scheme Compiler. User Manual for Version 2.9a*. December 2006.

[39] *Scheme Requests for Implementation*. February 2007. `http://srfi.schemers.org`.

[40] Richard M. Stallman: *GNU Emacs Manual*. January 2007. `http://www.gnu.org/software/emacs/manual/`.

[41] Guy Lewis Steele, Jr., Scott E. Fahlman, Richard P. Gabriel, David A. Moon, Daniel L. Weinreb, Daniel Gureasko Bobrow, Linda G. DeMichiel, Sonya E. Keene, Gregor Kiczales, Crispin Perdue, Kent M. Pitman, Richard Waters and Jon L White: Common Lisp. *The Language. Second Edition*. Digital Press. 1990.

[42] Philip Taylor, Jiři Zlatuška and Karel Skoupý: "The $\mathcal{NTS}$ Project: From Conception to Implementation". *Cahiers GUTenberg*, Vol. 35–36, pp. 53–77. May 2000.

[43] TUG Working Group on a TEX Directory Structure: *A Directory Structure for TEX Files*. `http://tug.org/tds`.

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

[45] Larry Wall, Tom Christiansen and Jon Orwant: *Programming Perl*. 3rd edition. O'Reilly & Associates, Inc. July 2000.

[46] Thomas Widman: "Bibulus—a Perl XML Replacement for BibTEX". In: *EuroTEX 2003*, pp. 137–141. ENSTB. June 2003.

[47] Jiři Zlatuška: "$\mathcal{NTS}$: Programming Languages and Paradigms". In: *EuroTEX 1999*, pp. 241–245. Heidelberg (Germany). September 1999.