Languages for Bibliography Styles

Jean-Michel Hufflen

Languages for bibliography styles

Jean-Michel Hufflen
LIFC (EA CNRS 4269)
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

BibTeX is the most commonly used bibliography processor in conjunction with LaTeX. To put bibliography styles into action, it uses a stack-based language written with postfixed notation. Recently, other approaches have been proposed: some use a script programming language for designing bibliography styles, e.g., Perl in Bibulus; some are based on converters to XML texts and use XSLT for bibliography styles; a recent proposal — the biblatex package — consists of using LaTeX commands to control the layout of generated references, and more. We propose a comparative study of these approaches and show which programming styles are encouraged, from a point of view related to methodology. Finally, we explain how this study has influenced the design of MlBibTeX, our multilingual reimplementation of BibTeX.

Keywords Bibliographies, bibliography styles, BibTeX, software quality, bst, BibTeX++, cl-bibtex, MlBibTeX, packages natbib, jurabib, and latexbib, Tib, XML, XSLT, nbst, Perl, DSSSL.

1 Introduction

Three decades ago,[1] some programming languages were designed to be universal, that is, to serve all purposes. All of these languages — e.g., PL/1 [25], Ada [2] — have failed to be accepted as filling this role. Nowadays, only the C programming language [30] is still used for a very wide range of applications. Present-day programming languages are very diverse and put different paradigms into action: procedural programming, object-oriented programming, functional programming, process-oriented programming, logic programming, ... In addition, most of these present languages are specialised, that is, designed for particular purposes, even if they are not formally limited to only one class of applications: two good examples are Java [28] and C# [39], originally designed for client-server applications. But, if you are building a standalone application using the object-oriented paradigm and are especially interested in the efficiency of the resulting program, it is well-known that a better choice is C++ [47], even if code generated by Java and C# compilers have greatly improved since their first versions.

The purpose of this article is neither a comparison of all programming languages — which would be impossible — nor an absolute comparison of several programming languages — which would not be of interest — but a comparative study of languages used to develop *bibliography styles*. BibTeX [42] is the bibliography processor most commonly used in conjunction with the LaTeX word processor [40], so most of the bibliography styles used for LaTeX texts are written using bst, the stack-based language of BibTeX [40, § 13.6]. But other proposals exist, based on other programming paradigms, and this article aims to study the advantages and drawbacks of these approaches. We will not discuss the typographical conventions ruling the typesetting of bibliographies — readers interested in this topic can consult manuals like [5, § 10], [6, §§ 15.54–15.76], [16, pp. 53–54] — but are interested only in the *development* of bibliography styles — from scratch or derived from other existing styles — and the expressive power of languages used to do that.

In Section 2, we recall the main quality factors of software, and show which factors are interesting from a point of view related to bibliography styles. Then BibTeX is studied in Section 3, including some modern use of this program. Other approaches are based on XML,[2] as shown in Section 4. This experience

[1] That is, at the time of TeX's first version... Let us recall that we are celebrating TeX's 30th birthday.

[2] E**X**tensible **M**arkup **L**anguage. Readers interested in an introductory book to this formalism can refer to [44].

of dealing with several ways to develop bibliography styles has influenced the design of MlBibTeX — for '**M**ulti**L**ingual BibTeX', our multilingual reimplementation of BibTeX [18]: we explain that in Section 5. Reading this article requires only a basic knowledge of BibTeX and a little experience about bibliography styles; we think that our examples should be understandable,[3] even if readers do not know thoroughly the languages used throughout this article.

2 Criteria

Of course, this section does not aim to replace a textbook about software quality, we just make precise the terminology we use. Then we explain how these notions are applied to bibliography styles.

2.1 General point of view

The main reference for the terminology used in software quality is the beginning of [38], as recalled in most works within this topic. [38, Ch. 1] clearly distinguishes *external* quality factors, that may be detected by users of a product, and *internal* factors, that are only perceptible to computer professionals. Here are the main external quality factors:

correctness the ability of software products to exactly perform their tasks, as defined by the requirements and specification;

robustness the ability of software systems to work even in abnormal conditions;

extendability the ease with which products may be adapted to changes of specifications;

reusability the ability of products may be combined with others;

others being *efficiency*, *portability*, *verifiability*, *integrity*, *ease of use*, etc. Internal quality factors include *modularity*, *legibility*, *maintainability*, etc. The factors related to modularity are studied more precisely in [38, Ch. 2], they include:

modular decomposability the ability for a problem to be decomposed into subproblems;

modular composability the ability for modules to be combined freely with each other;

modular understandability each module can be separately understood by a human reader;

modular continuity a small change in a problem specification results in a change of just a module or few modules.[4]

[3] Complete texts may be downloaded from MlBibTeX's home page: `http://lifc.univ-fcomte.fr/~hufflen/texts/mlbibtex/hc-styles/`.

[4] This terminology is related to mathematical analysis: a function is continuous if a small change in the argument will yield a small change in the result.

2.2 Tasks of a bibliography processor

Given *citation keys* — stored in an `.aux` file when a source text is processed by LaTeX [40, Fig. 12.1] — a bibliography processor searches bibliography database files for resources associated with these keys, performs a sort operation on bibliographical items,[5] and arranges them according to a bibliography style, the result being a source file for a 'References' section, suitable for a word processor. So does BibTeX.

Roughly speaking, a bibliography has to do two kinds of tasks:

- some are related to 'pure' programming, e.g., sorting bibliographical items, while
- others are related to put markup, in order for the word processor to be able to typeset the bibliography of a printed work.

The extendability of such a tool concerns these two kinds of tasks. On the one hand, we should be able to add a new relation order for sorting bibliographical items, since these lexicographical orders are language-dependent [24]. On the other hand, we should be able to build a new bibliography style, according to a publisher's specification. This style may be developed from scratch if we do not find a suitable existing style. Or we can get it by introducing some changes to another style, i.e., *reusing* some parts of the previous style. In addition, finding the parts that have to be changed is related to the notion of modular understandability. Of course, building a new bibliography style is not an end-user's task, but it should be possible by people other than the bibliography processor's developers.

Another notion is related to extending a bibliography processor: improving it so that it is usable with more word processors. If we consider the formats built on TeX [34], LaTeX is still widely used, but more and more people are interested in alternatives, such as ConTeXt [13]. Likewise, some new typeset engines, such as XeTeX [32] or LuaTeX [14], should be taken into account. In addition, it should be possible to put the contents of a bibliography database file on a Web page, that is, to express the information about these items using the HTML language.[6] A last example is given by RTF:[7] at first glance, deriving bibliographies using the internal markup language of Microsoft Word may seem strange, but

[5] ... unless the bibliography style is *unsorted*, that is, the order of items is the order of first citations. In practice, most bibliography styles are 'sorted'.

[6] **H**yper**T**ext **M**arkup **L**anguage. [41] is a good introduction to this language.

[7] **R**ich **T**ext **F**ormat. A good introductory book to this markup language is [4].

```
@STRING{srd = {Stephen Reeder Donaldson}}

@BOOK{donaldson1993,
        AUTHOR = srd,
        TITLE = {The Gap into Power: A Dark and
                 Hungry God Arises},
        PUBLISHER = {HarperCollins},
        SERIES = {The Gap},
        NUMBER = 3,
        YEAR = 1993}

@BOOK{donaldson1993a,
        EDITOR = srd,
        TITLE = {Strange Dreams},
        PUBLISHER = {Bantam-Spectra},
        YEAR = 1993}

@BOOK{murphy-mullaney2007,
        AUTHOR = {Warren Murphy and James
                  Mullaney},
        TITLE = {Choke Hold},
        PUBLISHER = {Tor},
        ADDRESS = {New-York},
        SERIES = {The New Destroyer},
        NUMBER = 2,
        NOTE = {The original series has been
                created by Richard Sapir and
                Warren Murphy},
        YEAR = 2007,
        MONTH = nov}
```

Figure 1: Bibliographical entries in the .bib format.

such a strategy may cause Word end-users to discover progressively the tools related to TEX.

3 BibTEX

3.1 Basic use

How to use BibTEX in conjunction with LATEX is explained in [40, § 12.1.3], and the .bib format, used within bibliography database files, is detailed in [40, § 13.2]; an example is given in Figure 1. As mentioned above, bibliography styles are written in a stack-based language using postfixed notation. As an example, Figure 2 gives two functions used within the plain style of BibTEX.

BibTEX is indisputably correct[8] and robust: as far as we have used it, the bibliographies it derives have satisfactory layout, at least for bibliographies of English-language documents. In addition, it has never crashed during our usage of it, even when dealing with syntactically incorrect .bib files.

[8] When the word 'correct' is used in software engineering, it is related to the existence of a formal specification — i.e., a mathematical description — of the behaviour, and the program should have been proved correct w.r.t. this specification. Here we adopt a more basic and intuitive sense: the program's results are what is expected by end-users.

```
FUNCTION {format.title}
{ title empty$
    { "" }
    { title "t" change.case$ }
  if$
}

FUNCTION {new.sentence.checkb}
{ empty$
  swap$ empty$
  and
    'skip$
    'new.sentence
  if$
}
```

Figure 2: Two functions from BibTEX's plain style.

Extending BibTEX, however, may be very tedious, especially for functionalities related to *programming*. For example, the only way to control the SORT command consists of using the entry variable `sort.key$` [40, Table 13.7]. Some workarounds may allow the definition of sort procedures according to lexicographic orders for natural languages other than English, but with great difficulty. Developing bibliography styles for word processors other than LATEX has been done, but only for formatters built on TEX, e.g., ConTEXt [17]. In other cases, this task may be difficult since some features related to TEX are hardwired in some built-in functions of BibTEX, e.g., the use of '~' for an unbreakable space character is in the specification of the `format.name$` function [23]. As an example, there is a converter from .bib format to HTML: BibTEX2HTML [9]. It uses BibTEX, but most of this translator is not written using BibTEX's language, but in Objective CAML,[9] a strongly typed functional programming language including object-oriented features [37]. Using such a tool — as well as the bibliography styles developed for ConTEXt's texts [17] — is possible only if end-users do not put LATEX commands inside the values associated with BibTEX's fields.

We think that the continuity of the bibliography styles written using the bst language is average. Introducing some changes concerning the layout of fragments is easy, e.g., short-circuiting case changes for a title, as shown in [40, § 13.6.3], as well as changing the style of a string by using a command like '`\emph{...}`'. That is due to the fact that inserting additional strings before or after the contents of a field is easy if this information is at the stack's top and has not been popped yet by means of the `write$`

[9] **C**ategorical **A**bstract **M**achine **L**anguage.

```
\bibitem[{Murphy\jbbtasep Mullaney\jbdy {2007}}%
         {}%
         {{0}{}{book}{2007}{}{}{}{}%
          {New-York\bpubaddr {} Tor\bibbdsep {} 2007}}%
         {{Choke Hold}%
          {}{}{2}{}{}{}{}{}}%
        ]{murphy-mullaney2007}
 \jbbibargs {\bibnf {Murphy} {Warren} {W.} {} {}\Bibbtasep \bibnf {Mullaney}
  {James} {J.} {} {}} {Warren MurphyJames Mullaney} {aus} {\bibtfont {Choke
  Hold}\bibatsep\  \apyformat {New-York\bpubaddr {} Tor\bibbdsep {} \novname\
  2007} \numberandseries {2}{The New Destroyer Series} \jbnote {1} {The
  original series has been created by Richard Sapir and Warren Murphy} }
  {\bibhowcited}  \jbendnote {The original series has been created by Richard
  Sapir and Warren Murphy} \jbdoitem {{Murphy}{Warren}{W.}{}{};
  {Mullaney}{James}{J.}{}{}} {} {} \bibAnnoteFile {murphy-mullaney2007}
```

Figure 3: BibTeX's output as used by the jurabib package.

function. For the same reason, adding a closing punctuation sign is easy; a shorthand example to do that is the `add.period$` function. Often handling a new field is easy, too [40, § 13.6.3]. On the other hand, changing the order of appearance of fields may be tedious.

In addition, it is well-known that there is no modularity within the bst language: each style is a monolithic file. If you develop a new style from an existing one, you just copy the .bst file onto a new file, and apply your changes. Of course, doing such a task requires good 'modular understanding' of the functions belonging to the 'old' style. Sometimes, that is easy — cf. the `format.title` function given in Figure 2 — while other times, understanding the role of a function is possible only if you know the stack's state — cf. the `new.sentence.checkb` function in the same figure.[10]

3.2 Task delegation

Originally, all the predefined bibliography styles provided by BibTeX's generated 'pure' LaTeX texts, in the sense that only basic LaTeX commands were used: the `\bibitem` command, the `thebibliography` environment [40, § 12.1.2], and some additional commands for word capitalisation or emphasis. No additional package was required when derived bibliographies were processed by LaTeX.

This situation has changed when the author-date system was implemented by the natbib package and the bibliography styles associated with it [40, § 12.3.2]. Progressively, other bibliography styles have been released, working as follows: BibTeX's output is marked up with LaTeX commands defined in an additional package. Citation and formatting functions can be customised by redefining these commands. In other words, we can say that BibTeX *delegates* the layout of bibliographies to these commands.

3.2.1 Interface packages

Figure 3 gives an example of using the jurabib bibliography style. The LaTeX commands provided by the jurabib package can be redefined like any LaTeX command, although the best method is to use the `\jurabibsetup` command, as shown in [40, § 12.5.1]. A similar approach is used within the amsxport bibliography style [8] and the bibliography styles usable with ConTeXt [17].

This *modus operandi* is taken to extremes by the biblatex package [36]. In such cases, BibTeX is used only to search bibliography database files and sort references. The advantage: end-users can customise the layout of bibliographies without any knowledge of the bst language. But BibTeX still remains used to sort references, and this task is not easily customisable, as mentioned above.

3.2.2 Tib

In fact, this notion of task delegation already existed in Tib [1], a bibliography processor initially designed for use with Plain TeX, although it can also be used with LaTeX. An example of a bibliography style file used by Tib is given in Figure 4: it consists of some Tib commands — e.g., '`f`' for 'citations as footnotes' — followed by some definitions of TeX commands for typesetting citation references and bibliographies' items. That is, Tib delegates a bibliography's layout to these commands. Let us recall that the bibliography database files searched by this processor do not

[10] This function is used when the decision of beginning a new sentence within a reference depends on the presence of two fields within an entry.

```
#
# standard footnote format (latex)
#
# if titles are desired in loc. cit. references, see note in stdftl.ttx
#
#         include word-definition file (journals and publishers)
I TMACLIB amsabb.ttz
f         footnotes
L         use ibid and loc cit
C0        empty citation string
O         for multiple citations use ordering of reference file

%The lines below are copied verbatim into the output document as TeX commands.
%First the file Macros.ttx is \input with Macros and default settings.
%The control string \TMACLIB is just a path.
%The \footnote macro is from LaTeX
%
\input \TMACLIB stdftl.ttx %macros for formatting reference list
\Refstda\Citesuper %set general formats for reference list and citations
\def\Lcitemark{\footnotemark}\def\Rcitemark{}
\def\Citecomma{$^,$\footnotemark}
\def\LAcitemark{\addtocounter{footnote}{1}\arabic{footnote}}
\def\RAcitemark{}
\def\LIcitemark#1\RIcitemark{\def\Ztest{ }\def\Zstr{#1}}
```

Figure 4: The footl.tib file.

```
%A |srd|
%T The Gap into Power: A Dark and Hungry God
Arises
%P HarperCollins
%S The Gap
%N 3
%D 1993

%E |srd|
%T Strange Dreams
%I Bantam-Spectra
%D 1993

%A Warren Murphy
%A James Mullaney
...
%O November 2007. The original series...
```

The 'srd' abbreviation should be defined by means of the following Tib command:

```
D srd Stephen Reeder Donaldson
```

Figure 5: Entries using the Refer format.

use the .bib format, but rather the Refer format,[11] an example being given in Figure 5.

3.3 Extending bst

The following works allow bibliography style writers to compile bst styles, and annotate or extend the result. As far as we know, they are not widely used. If we consider a style already written in bst and to be adapted, this approach allows more ambitious changes. However, they do not propose a new methodology for designing such styles, so taking maximum advantage of the target languages is difficult for style designers.

3.3.1 BibTeX++

BibTeX++ [31] allows a bst style file to be compiled into Java classes [28]. As an example, the `new.sentence.checkb` function (cf. Fig. 2) is compiled into the Java function `new_sentence_checkb` given in Figure 6. BibTeX++ can also run native bibliography styles developed in Java, from scratch or derived from the compilation of 'old' styles. Other functionalities, such as the production of references for programs other than LaTeX, can be implemented by means of *plug-ins*. There are six steps in BibTeX++'s process: for example, parsing a .bst file is the fourth one. After each step, there is a *hook*, as a callback that allows this process to be customised.

3.3.2 cl-bibtex

cl-bibtex [35] is based on ANSI[12] Common Lisp [11]. It includes an interpreter for the bst language, and

[11] The pybibliographer program can be used as a converter from the .bib format to the Refer format: see [40, § 13.4.5] for more details.

[12] **A**merican **N**ational **S**tandards **I**nstitute.

```
public void new_sentence_checkb(String s0,
                                String s1) {
  int i0, i1 ; i1 = BuiltIn.empty(s1) ;
  i0 = i1 ; i1 = BuiltIn.empty(s0) ;
  i0 = and(new Cell(i0),i1) ;
  if (i0 <= 0) new_sentence() ;
}
```

Figure 6: A bst function compiled into Java.

```
(define-bst-primitive "if$"
  ((pred (boolean)) (then (symbol body))
   (else (symbol body)))
  ()
  :interpreted
  (bst-execute-stack-literal
   (if pred then else)))
```

Figure 7: Implementation of `if$` in cl-bibtex.

can also compile a BibTeX style file into a Common Lisp program, as a starting point for customising such a style, by refining the corresponding Common Lisp program. As a short example, we show in Figure 7 how the `if$` function of BibTeX is implemented.

4 Using XML-like formats

Over the past several years, XML has become a central formalism for data interchange, so some projects are based on an XML-like language representing bibliographical items.

4.1 Converters

Several converters from the .bib format into an XML-like format have been developed: the bib2xml program [43], and the converter used as part of the BibTeXXML project [12]. MlBibTeX uses such a converter, too, and the result of the conversion of the second bibliographical entry of Figure 1 is given in Figure 8; the conventions used throughout such XML texts are a revision of the specification given in [10, § B.4.4].

The main difficulty of these tools is related to the LaTeX commands put inside the values associated with BibTeX fields. The bib2xml converter expands the commands for accents and diacritical signs into the corresponding single letters belonging to the Unicode encoding [48], but just drops out the '\' characters that open the other commands. MlBibTeX's converter processes more commands — e.g., `\emph`, `\textbf` — but of course, the way of dealing with user-defined commands should be defined by end-users [21].

```
<book id="donaldson1993a">
  <editor>
    <name>
      <personname>
        <first>Stephen Reeder</first>
        <last>Donaldson</last>
      </personname>
    </name>
  </editor>
  <title>Strange Dreams</title>
  <publisher>Bantam Spectra</publisher>
  <year>1993</year>
</book>
```

Figure 8: XML-like format used in MlBibTeX.

4.2 XSLT

XSLT[13] is the language used for the transformation of XML texts. Building a 'References' section is a particular case of transformation. This point is true for LaTeX source files as well as verbatim texts or HTML pages. Figure 9 shows how multiple authors or editors connected by an empty `and` tag can be processed, the result being a source text for LaTeX. More ambitious examples of using XSLT for typesetting texts are given in [46].

We have personally written many XSLT programs serving very diverse purposes. This language allows good modularity and reusability of fragments of existing programs. It allows users to write robust programs, too. As for developing bibliography styles, it offers good continuity, except for multilingual extensions. It was difficult to add information for a natural language without directly modifying an existing style. More precisely, that was difficult with the first version (1.0) [49], but has been improved in XSLT 2.0 where using *modes* has been refined [50, § 6.5]. Likewise, the expressive power of the `xsl:sort` element has been improved in this new version [50, § 13].

Extending XSLT functionalities often consists of calling external functions written using a more 'classical' programming language such as C or Java. That is possible, but not in a portable way, because it depends on the programming languages accepted by each XSLT processor. In practice, this point mainly concerns new lexicographical order relations within bibliography styles.

4.3 nbst

nbst[14] is the language used within MlBibTeX for specifying bibliography styles. As explained in [18],

[13] e**X**tensible **S**tylesheet **L**anguage **T**ransformations.
[14] **N**ew **B**ibliography **ST**yles.

```
<xsl:template match="author">
  <xsl:apply-templates/><xsl:text>. </xsl:text>
</xsl:template>

<xsl:template match="editor">
  <xsl:apply-templates/>
  <xsl:text>, </xsl:text>
  <xsl:choose>
    <xsl:when test="count(*) &gt; 1">
      <xsl:text>\bbled</xsl:text>
    </xsl:when>
    <xsl:otherwise>\bbleds</xsl:otherwise>
  </xsl:choose>
  <xsl:text>. </xsl:text>
</xsl:template>

<xsl:template match="name | personname">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="first">
  <xsl:value-of select="concat(.,' ')"/>
</xsl:template>

<xsl:template match="last">
  <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="and">
  <xsl:choose>
    <xsl:when
      test="following-sibling::and or
            following-sibling::and-others">
      <xsl:text>, </xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text> \bbland\ </xsl:text>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Figure 9: Dealing with authors or editors in XSLT.

this language is close to XSLT, and introduces a kind of inheritance for natural languages' specification. First, we look for a template whose `language` attribute matches the current language, and second a more general template, without the `language` attribute.

MlBibTeX is written in Scheme [29], and XML texts are represented using the SXML[15] format [33]. Roughly speaking, this format uses prefixed notation, surrounded by parentheses — as in any Lisp dialect — for tags surrounding contents. As an example, the result of parsing the bibliographical entries of Figure 1 is sketched in Figure 10. Dealing directly with Scheme functions is needed when new language-dependent lexicographical order relations are to be

[15] Scheme implementation of XML.

```
(*TOP*
 (*PI* xml "version=\"1.0\"
       encoding=\"ISO-8859-1\"")
 (mlbiblio
  ...
  (book
   (@ (id "murphy-mullaney2007"))
   (author
    (name (personname (first "Warren")
                      (last "Murphy")))
    (and)
    (name (personname (first "James")
                      (last "Mullaney"))))
   (title "Choke Hold") (publisher "Tor")
   (year "2007") (month (nov)) (number "2")
   (series "The New Destroyer")
   (address "New-York")
   (note "The original series..."))))
```

Figure 10: Using the SXML format.

added [24]. nbst texts can call functions directly written in Scheme, as well.

4.4 Perl

Perl[16] [51] can be used for bibliography styles, as is done by Bibulus [52], this program being based on the bib2xml converter [43]. The resulting bibliography styles are compact, modular, and easily extensible. The modularity of Bibulus styles can be illustrated by the `\bibulus` command that can be used in place of the `\bibliographystyle` command:

```
\bibulus{citationstyle=numerical,
         surname=comes-first,
         givennames=initials,
         blockpuctuation=.}
```

Multilingual features are processed by means of substitutions, which can easily be incorrect: for example, a month name precedes the year in English, but follows the year in Hungarian. So a rough substitution of an English month name is insufficient.[17] Last but not least, Bibulus is not very easy to use, it is presently accessible only to developers.

4.5 DSSSL

DSSSL[18] [27] was initially designed as the stylesheet language for SGML[19] texts. Since XML is a subset of SGML, stylesheets written using DSSSL can be applied to XML texts. DSSSL is rarely used now,

[16] **P**ractical **E**xtraction **R**eport **L**anguage.

[17] The same criticism holds for the babelbib package [15].

[18] **D**ocument **S**tyle **S**emantics **S**pecification **L**anguage.

[19] **S**tandard **G**eneralised **M**arkup **L**anguage. Now it is only of historical interest. Readers interested in this metalanguage can refer to [3].

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN">
<style-sheet>
  <style-specification id="hcs">
    <style-specification-body>
      (root (let ((margin-size 1in))   ; DSSSL uses quantities, analogous to TeX's dimensions.
              (make simple-page-sequence
                ;; An identifier ending with the ':' characters is a key, bound to the following value.
                page-width: 210mm page-height: 297mm left-margin: margin-size
                right-margin: margin-size top-margin: margin-size bottom-margin: margin-size
                header-margin: margin-size footer-margin: 12mm center-footer: (page-number-sosofo)
                (process-children))))
      (element book
        (make-reference (lambda (current-children) ; Function to be applied as soon as the general
                          (make sequence           ; framework for a reference has just been built: cf. the
                            (author-xor-editor current-children) ; definition of our make-reference
                            (process-matching-children "title")  ; function below.
                            (process-seriesinfo current-children)
                            (apply sosofo-append
                                   (map process-matching-children
                                        '("publisher" "address" "month" "year" "note")))))))
      (element author (process-author-or-editor))   ; The same for editor elements.
      (element name (process-children-trim))   ; The same for number elements.
      (element personname (processing-matching-children "first" "von" "last" "junior"))
      (element first (ending-with space-literal))
      (element last (process-children-trim))
      (element and (if (node-list-empty? (select-elements (follow (current-node)) "and"))
                       (literal " and " )
                       comma-space-literal))
      (element (book title) (make sequence font-posture: 'italic (process-and-closing-period)))
      (element year (process-and-closing-period))   ; The same for series and note elements.
      (element month (make sequence (process-children) space-literal))
      (element jan (literal "January")) ...  ; Other month elements skipped.
      (element publisher (ending-with comma-space-literal))   ; The same for address elements.
      ...
      ;; Definitions for particular literals and strings:
      (define comma-space-literal (literal ", "))
      (define period-string ".")
      (define space-literal (literal " "))
      ;; General framework for references' layout:
      (define make-reference
        (let ((biblioentry-indent 20pt))
          (lambda (process-f)
            (make paragraph
              first-line-start-indent: (- biblioentry-indent) font-size: 12pt quadding: 'justify
              space-before: 10pt start-indent: biblioentry-indent
              (literal "[" (attribute-string "id") "] ") (process-f (children (current-node)))))))
      ;; Some utility functions:
      (define (process-author-or-editor)
        (process-matching-children "name" "and"))
      (define (ending-with literal-0)
        (make sequence (process-children-trim) literal-0))
      ...
    </style-specification-body>
  </style-specification>
</style-sheet>
```

Figure 11: Example of a DSSSL stylesheet.

```
(define (process-and-closing-period)
  (let ((the-string (string-trim-right (data (current-node)))))    ; Get the contents and leave trailing
                                                                   ; space characters.
    (literal (if (check-for-closing-sign? the-string)    ; Checking if the-string ends with '.', '?', or '!'.
                 the-string
                 (string-append the-string period-string)))))

(define (author-xor-editor node-list)
  (let ((author-node-list (select-elements node-list "author"))
        (editor-node-list (select-elements node-list "editor")))
    (make sequence
      (cond ((node-list-empty? author-node-list)
             (if (node-list-empty? editor-node-list)
                 (error "Neither author, nor editor!")
                 (make sequence (process-node-list editor-node-list) (literal ", editor."))))
            ((node-list-empty? editor-node-list)
             (make sequence
               (process-node-list author-node-list)
               (if (check-for-closing-sign? (string-trim-right (data author-node-list)))
                   (empty-sosofo)
                   (literal period-string))))
            (else (error "Both author and editor!")))
      (literal " "))))
```

Figure 12: Some auxiliary functions implemented in DSSSL.

```
(define (b-if$ sxml-mlbiblio-tree current-entry-plus)
  ;; sxml-biblio-tree is the complete tree of all the entries to be processed, current-entry-plus the annotated
  ;; tree of the current entry.
  (let* ((i2 ((b-bst-stack-pv 'pop)))    ; "Else" part.
         (i1 ((b-bst-stack-pv 'pop)))    ; "Then" part.
         (i0 ((b-bst-stack-pv 'pop))))   ; Condition.
    (if (integer? i0)
        (b-process-sequence (if (positive? i0) i1 i2) sxml-mlbiblio-tree current-entry-plus)
        (begin
          ((msg-manager 'bst-type-conflict) 'if$ i0)
          #t))))
```

Figure 13: Implementing `if$` within MlBibTeX's compatibility mode.

but the example we show illustrates how a functional programming language can implement a bibliography style. More examples can be found in [10, § 7.5].

Figure 11 gives some excerpts of a stylesheet that displays the items of a bibliography by labelling them with their own keys. The core expression language of DSSSL is a side-effect free subset of Scheme. As shown in Figure 11, processing elements uses pattern-matching:

```
(element name E)
(element (name0 name) E0)
```

the *E* expression specifies how to process the *name* element, unless this element is a child of the $name_0$ element, in which case the E_0 expression applies. The choice of the accurate expression is launched by functions such as `process-matching-children`, `process-children`, and `process-node-list`.

Expressions like *E* or E_0 consist of assembling *literals* by means of the `make` form, using types predefined in DSSSL: `paragraph`, `sequence`, ... The generic type of such results is called *sosofo*[20] w.r.t. DSSSL's terminology.

Figure 12 illustrates this style of programming by showing some specific details: how to implement BibTeX's `add.period$` function, and the switch between `author` and `editor` elements for a book. This stylesheet can be run by the jade[21] program; as shown in [10, § 7.5.2], the TeX-like typeset engine able to process such results is JadeTeX.

Fragments of DSSL stylesheets can be organised into libraries, so this language is modular. Most of the implementations of it are robust, efficient, but

[20] **S**pecification **O**f a **S**equence **O**f **F**low **O**bjects.
[21] **J**ames Clark's **A**wesome **D**SSSL **E**ngine.

they are neither extensible, nor easy to use, because we have to make precise a predefined *backend*. For example, if jade is used to process the complete stylesheet given in Figures 11 & 12, the possible backends are `tex` (resp. `rtf`), in which case the result is to be processed by JadeTEX (resp. Microsoft Word or OpenOffice). Deriving texts directly processable by LATEX or ConTEXt is impossible.

5 The application to MlBibTEX

When we designed MlBibTEX's present version, we had had much experience in programming DSSSL and XSLT stylesheets. We thought that a language close to DSSSL would provide more expressive power for developing, but would be accessible only for programmers. A language close to XSLT is better from this point of view, provided that an extension mechanism is given for operations related to pure programming, e.g., the definition of new relation orders [24]. In addition, performing some operations may be more difficult than in bst, e.g., the `add.period$` functionality.

The only solution is to provide an initial library *legible* from a point of view related to methodology [19]. A compatibility mode is needed in order to ease the transition between old and new bibliography styles [20] — Figure 13 shows how the `if$` function is implemented within this mode, in comparison with the implementation of cl-bibtex, given in Figure 7. This progressively led us to the architecture described in [22].

We can be objective about MlBibTEX only with difficulty. However, several points seem to us to confirm our choices. First, XSLT has succeeded as a language able to deal with XML texts, much more so than DSSSL with SGML texts. Second, the need for a classical programming language: using the whole expressive power of Scheme — and not a subset as in DSSSL — allowed us to program efficiently, by using advanced features of Scheme. Third, our experience with ConTEXt [21] seems to confirm the extendability of our tool.

6 Conclusion

Table 1 summarises our experience with the languages we have described above. Of course, this synthesis is not as objective as benchmarks would have been. It is just a study of the effort we have made for developing bibliography styles, and a professional view of the results we have found.

To end, let us make a last remark about what is done in MlBibTEX: the separation of functionalities related to programming, written in Scheme, and specifications of layout, given in an XSLT-like language. Analogous combinations exist, the most widely used are a logic programming language, like Prolog,[22] called within a C (or similar) program. This *modus operandi* allows programmers to use a very specialised language only when it is suitable. There is an analogous example within TEX's world: LuaTEX. Functionalities related to typesetting are performed by commands built into TEX, whereas other functions are implemented by means of the Lua language [26]. So TEX is used as the wonderful typesetting engine that it is, and functionalities difficult to implement with TEX's language[23] are delegated to a more traditional programming language.

	BibTEX	XSLT	DSSSL	Bibulus
Correctness	✔			
Robustness	✔	✔	✔	
Extendability	✘	✔	✘	✔/✘
Reusability	✔/✘			
Modularity	poor	✔	✔	✔
Continuity	average	✔[a]	✔	✔
Efficiency	✔		✔	
Ease of use	✔	average	✘	✘

[a] . . . except for multilingual features, in XSLT 1.0.

Table 1: Languages for bibliography styles: synthesis.

BibTEX is still a powerful bibliography processor, but the main way to extend it easily concerns the layout of the bibliographies. That was sufficient some years ago, but not now with the use of Unicode, new processors like XeTEX, and new languages like HTML.

7 Acknowledgements

I have been able to write this article because I have had much occasion to become familiar with the languages and applications mentioned above. For example, some years ago, a colleague asked me to help him with a DSSSL program because I knew Scheme: that was my first contact with this language and SGML, before XML came out. Another time, a friend who used Plain TEX asked me a question about Tib, although I had merely heard of the name of this program, and I discovered it in this way. So I was thinking about all these people when I was writing this article, and I am grateful to them. Many thanks to Karl Berry: as usual, he is a conscientious proofreader and 'figure-positioner'.

[22] A good introductory book to this language is [7].
[23] Some examples can be found in [45].

References

[1] James C. Alexander: *T1b: a TEX Bibliographic Preprocessor*. Version 2.2, see CTAN:biblio/tib/tibdoc.tex. 1989.

[2] ANSI: *The Programming Language Ada® Reference Manual*. Technical Report ANSI/MIL-STD-1815A-1983, American National Standard Institute, Inc. LNCS No. 155, Springer-Verlag. 1983.

[3] Neil Bradley: *The Concise SGML Companion*. Addison-Wesley. 1997.

[4] Sean M. Burke: *RTF Pocket Guide*. O'Reilly. July 2003.

[5] Judith Butcher: *Copy-Editing. The Cambridge Handbook for Editors, Authors, Publishers*. 3rd edition. Cambridge University Press. 1992.

[6] *The Chicago Manual of Style*. The University of Chicago Press. The 14th edition of a manual of style revised and expanded. 1993.

[7] William F. Clocksin and Christopher S. Mellish: *Programming in Prolog*. 5th edition. Springer-Verlag. 2003.

[8] Michael Downes: "The amsrefs LaTeX Package and the amsxport BibTeX Style". *TUGboat*, Vol. 21, no. 3, pp. 201–209. September 2000.

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

[10] Michel Goossens and Sebastian Rahtz, with Eitan M. Gurari, Ross Moore, and Robert S. Sutor: *The LaTeX Web Companion*. Addison-Wesley Longmann, Inc., Reading, Massachusetts. May 1999.

[11] Paul Graham: *ANSI Common Lisp*. Series in Artificial Intelligence. Prentice Hall, Englewood Cliffs, New Jersey. 1996.

[12] Vidar Bronken Gundersen and Zeger W. Hendrikse: *BibTeX as XML Markup*. January 2007. http://bibtexml.sourceforge.net.

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

[14] Hans Hagen: "LuaTeX: Howling to the Moon". *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, Vol. 23, pp. 63–68. April 2006.

[15] Harald Harders: „Mehrsprachige Literaturverzeichnisse: Anwendung und Erweiterung des Pakets babelbib". *Die TEXnische Komödie*, Bd. 4/2003, S. 39–63. November 2003.

[16] *Hart's Rules for Composers and Readers at the University Press*. Oxford University Press. 39th edition. 1999.

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

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

[19] Jean-Michel Hufflen: "Bibliography Styles Easier with MlBibTeX". In: *Proc. EuroTEX 2005*, pp. 179–192. Pont-à Mousson, France. March 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: "MlBibTeX Architecture". *ArsTEXnica*, Vol. 2, pp. 54–59. In GUIT 2006 meeting. October 2006.

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

[24] Jean-Michel Hufflen: "Managing Order Relations in MlBibTeX". *TUGboat*, Vol. 29, no. 1, pp. 101–108. EuroBachoTeX 2007 proceedings. 2007.

[25] IBM System 360: *PL/1 Reference Manual*. March 1968.

[26] Roberto Ierusalimschy: *Programming in Lua*. 2nd edition. Lua.org. March 2006.

[27] International Standard ISO/IEC 10179:1996(E): *DSSSL*. 1996.

[28] *Java Technology*. March 2008. http://java.sun.com.

[29] 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.

[30] Brian W. Kernighan and Dennis M. Ritchie: *The C Programming Language*. 2nd edition. Prentice Hall. 1988.

[31] Ronan Keryell: "BibTeX++: Towards Higher-Order BibTeXing". In: *EuroTeX 2003*, p. 143. ENSTB. June 2003.

[32] Jonathan Kew: "XeTeX in TeX Live and beyond". *TUGboat*, Vol. 29, no. 1, pp. 146–150. EuroBachoTeX 2007 proceedings. 2007.

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

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

[35] Matthias Köppe: *A BibTeX System in Common Lisp*. January 2003. `http://www.nongnu.org/cl-bibtex`.

[36] Philipp Lehman: *The biblatex Package. Programmable Bibliographies and Citations*. Version 0.7 (beta). December 2007. `http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/doc/biblatex.pdf`.

[37] 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`.

[38] Bertrand Meyer: *Object-Oriented Software Construction*. Series in Computer Science. Prentice Hall International. 1988.

[39] Microsoft Corporation: *Microsoft C# Specifications*. Microsoft Press. 2001.

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

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

[42] Oren Patashnik: *BibTeXing*. February 1988. Part of the BibTeX distribution.

[43] Chris Putnam: *Bibliography Conversion Utilities*. February 2005. `http://www.scripps.edu/~cdputnam/software/bibutils/bibutils.html`.

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

[45] Denis B. Roegel : « Anatomie d'une macro ». *Cahiers GUTenberg*, Vol. 31, p. 19–27. Décembre 1998.

[46] Bob Stayton: *DocBook—XSL. The Complete Guide*. 3rd edition. Sagehill Enterprises. February 2005.

[47] Bjarne Stroustrup: *The C++ Programming Language*. 2nd edition. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts. 1991.

[48] The Unicode Consortium: *The Unicode Standard Version 5.0*. Addison-Wesley. November 2006.

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

[50] W3C: *XSL Transformations (XSLT). Version 2.0*. W3C Recommendation. Edited by Michael H. Kay. January 2007. `http://www.w3.org/TR/2007/WD-xslt20-20070123`.

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

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