The PracTEX Journal, 2007, No. 3
Article revision 2007/08/28

Tools for Collaborative Writing of Scientific LaTeX Documents

Arne Henningsen

Abstract Collaborative writing of documents requires a strong synchronisation among authors. This paper describes a possible way to organise the collaborative preparation of scientific LaTeX documents. The presented solution is primarily based on the version control system **Subversion**. The paper describes how **Subversion** can be used together with several other software tools and LaTeX packages to organise the collaborative preparation of LaTeX documents.

1 Introduction

Many scientific articles, reports, and books are written by more than one author. The collaborative preparation of documents requires a considerable amount of coordination among the authors. This coordination can be organised in many different ways, where the best way depends on the specific circumstances.

In this paper, I describe how the collaborative writing of LaTeX documents is organised at our department[1]. I present our software tools, and describe how we use them. Thus, this paper provides some ideas and hints that will be useful for other LaTeX users who prepare documents together with their co-authors.

2 Interchanging Documents

There are many ways to interchange documents among authors. One possibility is to compose documents by interchanging e-mail messages. This method has the advantage that common users generally do not have to install and learn the usage of any extra software, because virtually all authors have an e-mail account.

1. Division of Agricultural Policy, Department of Agricultural Economics, University of Kiel, Germany.

Furthermore, the author who has modified the document can easily attach the document and explain the changes by e-mail as well. Unfortunately, there is a problem when two or more authors are working, at the same time, on the same document. So, how can authors synchronise these files?

A second possibility is to provide the document on a common file server, which is available in most departments. The risk of overwriting each others' modifications can be eliminated by locking files that are currently edited. However, generally the file server can be only accessed from within a department. Hence, authors, who are out of the building, cannot use this method to update/commit their changes. In this case, they will have to use another way to contour this problem. So, how can authors access these files?

A third possibility is to use a version control system. A comprehensive list of version control systems can be found at [32]. Version control systems keep track of all changes in files in a project. If many authors modify a document at the same time, the version control system tries to merge all modifications automatically. Only if two (or more) authors have modified the same line, the modifications cannot be merged automatically, but the user has to resolve this "conflict" by deciding manually, which of the two changes should be kept. Authors can also comment their modifications so that the co-authors can easily understand the workflow of this file. As version control systems generally communicate over the internet (e.g. through TCP/IP connections), they can be used from different computers with internet connection.[2] The internet is only used for synchronising the files. Hence, a permanent internet connection is not required. The only drawback of a version control system could be that a it has to be installed and configured.[3]

2. A restrictive firewall policy might prevent the version control system from connecting to the internet. In this case, the network administrator has to be asked to open the appropriate port.

3. A version control system is useful even if a single user is working on a project. First, the user can track (and possibly revoke) all previous modifications. Second, this is a convenient way to have a backup of the files on other computers (e.g. on the version control server). Third, this allows the user to easily switch between different computers (e.g. office, laptop, home).

3 The Version Control System Subversion

At our department, we decided to use the open source version control system **Subversion** [26]. This software is considered as an improvement of the popular version control system **CVS**. The **Subversion** (**SVN**) version control system is based on a central **Subversion** server that hosts the "repositories".[4] Each user has a local "working copy" of (a part of) a remote "repository". For instance, users can "update" changes from the repository to their working copy, "commit" changes from their own working copy to the repository, or (re)view the differences between working copy and repository.

To set up a **Subversion** version control system, the **Subversion** *server* software has to be installed on a (single) computer with permanent internet access.[5] It can run on many Unix, modern MS Windows, and Mac OS X platforms.

Users do not have to install the **Subversion** *server* software, but a **Subversion** *client* software. This is the unique way to access the "repositories" on the server. Besides the basic **Subversion** command-line client, there are several Graphical User Interface Tools (GUIs) and plug-ins for accessing the **Subversion** server (see [27]). Additionally, there are very good manuals about **Subversion** freely available on the internet (e.g. [3]).

At our department, we run the **Subversion** server on a **GNU-Linux** system, because most **Linux** distributions include it. In this sense, installing, configuring, and maintaining **Subversion** is a very simple task.

Most MS Windows users access the **Subversion** server by the **TortoiseSVN** client [29], because it provides the most usual interface for common users. Linux users usually use the **Subversion** command-line client or **eSvn** GUI [15] with **KDiff3** [7] for showing complex differences.

4. A Repository can be thought of as a library, where authors keep successive revisions of one or more documents. The version control systems acts as the librarian between the author and the repository. For instance, the authors can ask the librarian to get the latest version of their projects or to commit a new version to the librarian. [5, modified]

5. If this computer has no static IP address, one can use a service like DynDNS [6] to be able to access the server with a static hostname.

4 Hosting LaTeX files in Subversion

On our **Subversion** server, we have one repository for a common `texmf` tree. Its structure complies with the "TeX Directory Structure" guidelines (TDS, [21], see figure 1). This repository provides LaTeX classes, LaTeX styles, and BibTeX styles

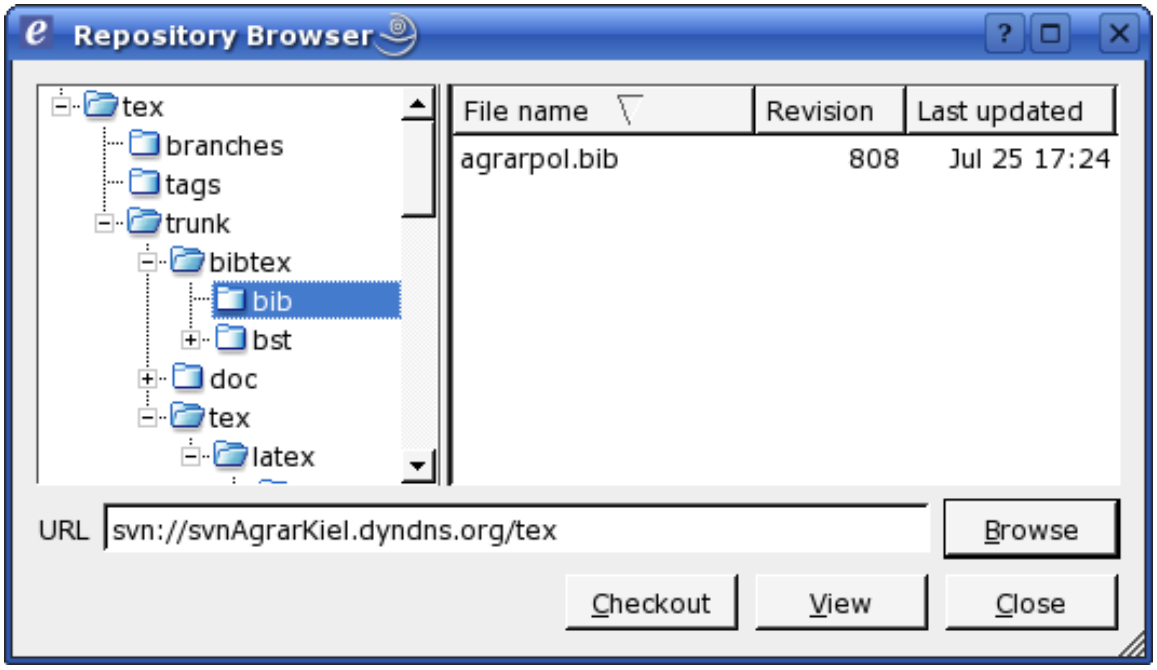

Figure 1: Common `texmf` tree shown in **eSvn**'s Repository Browser

that are not available in the LaTeX distributions of the users, e.g. because they were bought or developed for the internal use at our department. All users have a working copy of this repository and have configured LaTeX to use this as their personal `texmf` tree.[6] If a new class or style file has been added (but not if these files have been modified), the users have to update their "file name data base" (FNDB)

6. For instance, teTeX [8] users can edit their TeX configuration file (e.g. `/etc/texmf/web2c/texmf.cnf`) and set the variable `TEXMFHOME` to the path of the working copy of the common `texmf` tree (e.g. by `TEXMFHOME = $HOME/texmf`); MiKTeX [20] users can add the path of the working copy of the common `texmf` tree in the "Roots" tab of the MiKTeX Options.

before they can use these classes and styles.[7] Furthermore, the repository contains manuals explaining the specific LaTeX software solution at our department (e.g. this document).

The **Subversion** server hosts a separate repository for each project of our department. Although branching, merging, and tagging is less important for writing text documents than for writing source code for software, our repository layouts follow the recommendations of [3]. In this sense, each repository has the three directories `/trunk`, `/branches`, and `/tags`.

The most important directory is `/trunk`. If a single text document belongs to the project, all files and subdirectories of this text document are in `/trunk`. If the project yields two or more different text documents, `/trunk` contains a subdirectory for each text document. A slightly different version (a *branch*) of a text document (e.g. for presentation at a conference) can be prepared either in an additional subdirectory of `/trunk` or in a new subdirectory of `/branches`. When a text document is submitted to a journal or a conference, we create a *tag* in the directory `/tags` so that it is easy to identify the submitted version of the document at a later date. This feature has been proven very useful. When creating branches and tags, it is important always to use the **Subversion** client (and not the tools of the local file system) for these actions, because this saves disk space on the server and it preserves information about the same history of these documents.

Often the question arises, which files should be put under version control. Generally, all files that are directly modified by the user and that are necessary for compiling the document should be included in the version control system. Typically, these are the LaTeX source code (`*.tex`) files (the main document and possibly some subdocuments) and all pictures that are inserted in the document (`*.eps`, `*.jpg`, `*.png`, and `*.pdf` files). All LaTeX classes (`*.cls`), LaTeX styles (`*.sty`), BibTeX data bases (`*.bib`), and BibTeX styles (`*.bst`) generally should be hosted in the repository of the common `texmf` tree, but they could be included in the respective repository, if some (external) co-authors do not have access to the common `texmf` tree. On the other hand, all files that are automatically created or modified during the compilation process (e.g. `*.aut`, `*.aux`, `*.bbl`, `*.bix`, `*.blg`, `*.dvi`, `*.glo`, `*.gls`, `*.idx`, `*.ilg`, `*.ind`, `*.ist`, `*.lof`, `*.log`, `*.lot`, `*.nav`, `*.out`, `*.pdf`, `*.ps`, `*.snm`, and `*.toc` files) or by the (LaTeX or BibTeX) editor

7. For instance, teTeX [8] users have to execute `texhash`; MiKTeX [20] users have to click on the button "Refresh FNDB" in the "General" tab of the MiKTeX Options.

(e.g. `*.bak`, `*.bib~`, `*.kilepr`, `*.prj`, `*.sav`, `*.tcp`, `*.tmp`, `*.tps`, and `*.tex~` files) generally should be *not* under version control, because these files are not necessary for compilation and generally do not include additional information. Furthermore, these files are regularly modified so that conflicts are very likely.

5 **Subversion** really makes the **diff**erence

A great feature of a version control system is that all authors can easily trace the workflow of a project by viewing the differences between arbitrary versions of the files. Authors are primarily interested in "effective" modifications of the source code that change the compiled document, but not in "ineffective" modifications that have no impact on the compiled document (e.g. the position of line breaks). Software tools for comparing text documents ("diff tools") generally cannot differentiate between "effective" and "ineffective" modifications; they highlight both types of modifications. This considerably increases the effort to find and review the "effective" modifications. Therefore, "ineffective" modifications should be avoided.

In this sense, it is very important not to change the positions of line breaks without cause. Hence, automatic line wrapping of the users' LaTeX editors should be turned off and line breaks should be added manually. Otherwise, if a single word in the beginning of a paragraph is added or removed, all line breaks of this paragraph might change so that most diff tools indicate the entire paragraph as modified, because they compare the files line by line. The diff tools **wdiff** [10] and **dwdiff** [11] are not effected by the positions of line breaks, because they compare documents word by word.[8] However, their output is less clear so that modifications are more difficult to track.

A reasonable convention is to add a line break after each sentence and start each new sentence in a new line.[9] Furthermore, we split long sentences into

8. These tools cannot be used directly with the **Subversion** command-line switch `--diff-cmd`, but a small wrapper script has to be used. [1]

9. This also has an advantage beyond version control: if you want to find a sentence in your LaTeX code that you have seen in a compiled (DVI, PS, or PDF) file or on a printout, you can easily identify the first few words of this sentence and screen for these words on the left border of your editor window.

several lines so that each line has at most 80 characters,[10] because it is rather inconvenient to search for (small) differences in long lines. We find it very useful to introduce the additional line breaks at logical breaks of the sentence, e.g. before a relative clause or a new part of the sentence starts. An example LaTeX code that is formatted according to these guidelines is the source code of this document, which is available on PracTeX's website.

There is also another important reason for reducing the number of "ineffective" modifications: if several authors work on the same file, the probability that the same line is modified by two or more authors at the same time increases with the number of modified lines. Hence, "ineffective" modifications unnecessarily increase the risk of conflicts (see section 2).

Furthermore, version control systems allow a very effective quality assurance measure: all authors should critically review their own modifications before they commit them to the repository (see figure 2). The differences between the user's working copy and the repository can be easily inspected with a single **Subversion** command or with one or two clicks in a graphical **Subversion** client. Furthermore, authors should verify that their code can be compiled flawlessly before they commit their modifications to the repository. Otherwise, the co-authors have to pay for these mistakes when they want to compile the document. However, this directive is not only reasonable for version control systems but also for all other ways to interchange documents among authors.

Subversion has a feature called "Keyword Substitution" that includes dynamic version information about a file (e.g. the revision number or the last author) into the contents of the file itself [3, chapter 3]. Sometimes, it is useful to include these information not only as a comment in the LaTeX source code, but also in the (compiled) DVI, PS, or PDF document. This can be achieved with the LaTeX packages **svn** [16], **svninfo** [2], or **svn-multi** [19] (preferably).

The most important directives for collaborative writing of LaTeX documents with version control systems are summarised in box 1.

10. For instance, the LaTeX editor **Kile** [14] can assist the user in this task when it is configured to add a vertical line that marks the 80$^{\text{th}}$ column.

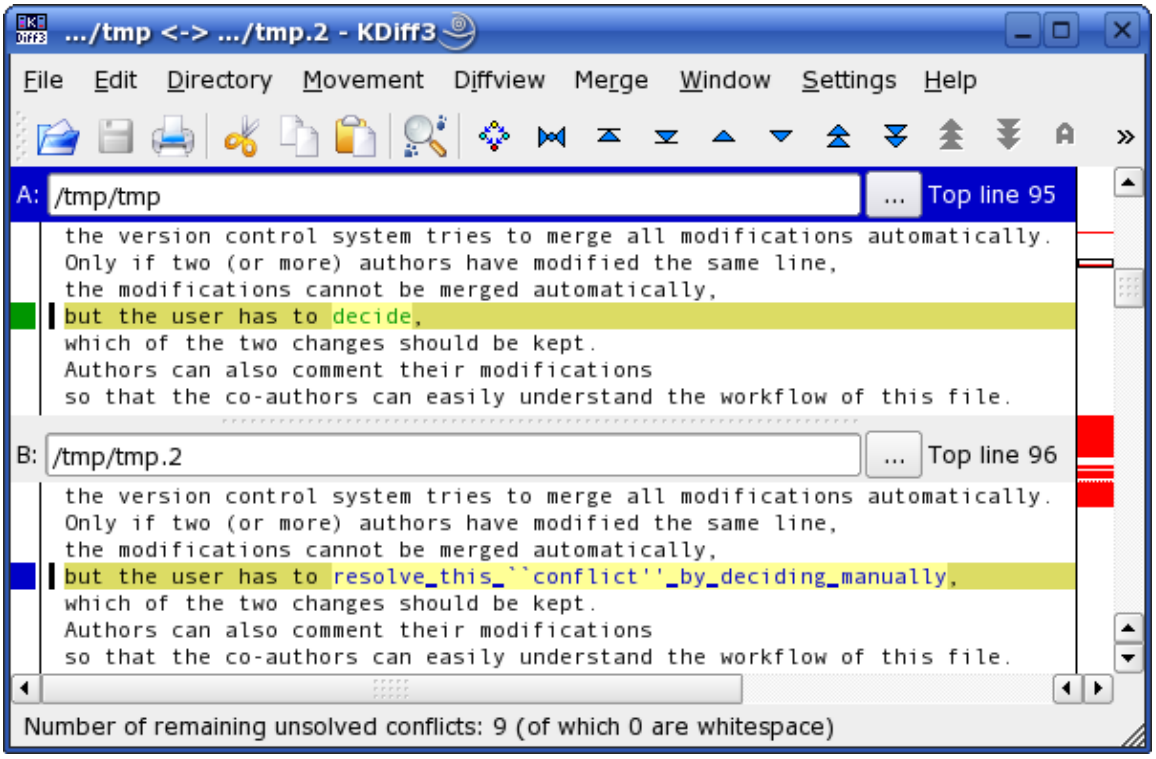

Figure 2: Reviewing modifications in **KDiff3**

6 Bibliography

Writing of scientific articles, reports, and books requires the citation of all relevant sources. BibTeX is an excellent tool for citing references and creating bibliographies [17, 9]. Many different BibTeX styles can be found on CTAN [23] and on the LaTeX Bibliography Styles Database [12]. If no suitable BibTeX style can be found, most desired styles can be conveniently assembled with **custombib**/**makebst** [4]. Furthermore, BibTeX style files can be created or modified manually; however this action requires knowledge of the (unnamed) postfix stack language that is used in BibTeX style files [18].

At our department, we have a common bibliographic data base in the BibTeX format (.bib file). It resides in our common `texmf` tree (see section 4) in the subdirectory `/bibtex/bib/` (see figure 1). Hence, all users can specify this bibliography

1. Avoid "ineffective" modifications.
2. Do not change line breaks without good reason.
3. Turn off automatic line wrapping of your LaTeX editor.
4. Start each new sentence in a new line.
5. Split long sentences into several lines so that each line has at most 80 characters.
6. Put only those files under version control that are directly modified by the user.
7. Verify that your code can be compiled flawlessly before committing your modifications to the repository.
8. Use **Subversion**'s diff feature to critically review your modifications before committing them to the repository.
9. Add a meaningful and descriptive comment when committing your modifications to the repository.
10. Use the **Subversion** client for copying, moving, or renaming files and folders that are under revision control.

Box 1: Directives for using LaTeX with version control systems

by only using the file name (without the full path) — no matter where the user's working copy of the common `texmf` tree is located.

All users edit our bibliographic data base with the graphical BibTeX editor **JabRef** [24]. As **JabRef** is written in **Java**, it runs on all major operating systems. As different versions of **JabRef** generally save files in a slightly different way (e.g. by introducing line breaks at different positions), all users should use the same (e.g. last stable) version of **JabRef**.[11]

11. Otherwise, there would be many differences between different versions of `.bib` files that solely originate from using different version of **JabRef**. Hence, it would be hard to find the real differences between the compared documents. Furthermore, the probability of conflicts would be much higher (see section 5).

JabRef is highly flexible and can be configured in many details. We make the following changes to the default configuration of **JabRef** to simplify our work. First, we specify the default pattern for BibTeX keys so that **JabRef** can automatically generate keys in our desired format. This can be done by selecting `Options` → `Preferences` → `Key pattern` and modifying the desired pattern in the field `Default pattern`. For instance, we use `[auth:lower][shortyear]` to get the last name of the first author in lower case and the last two digits of the year of the publication (see figure 3).

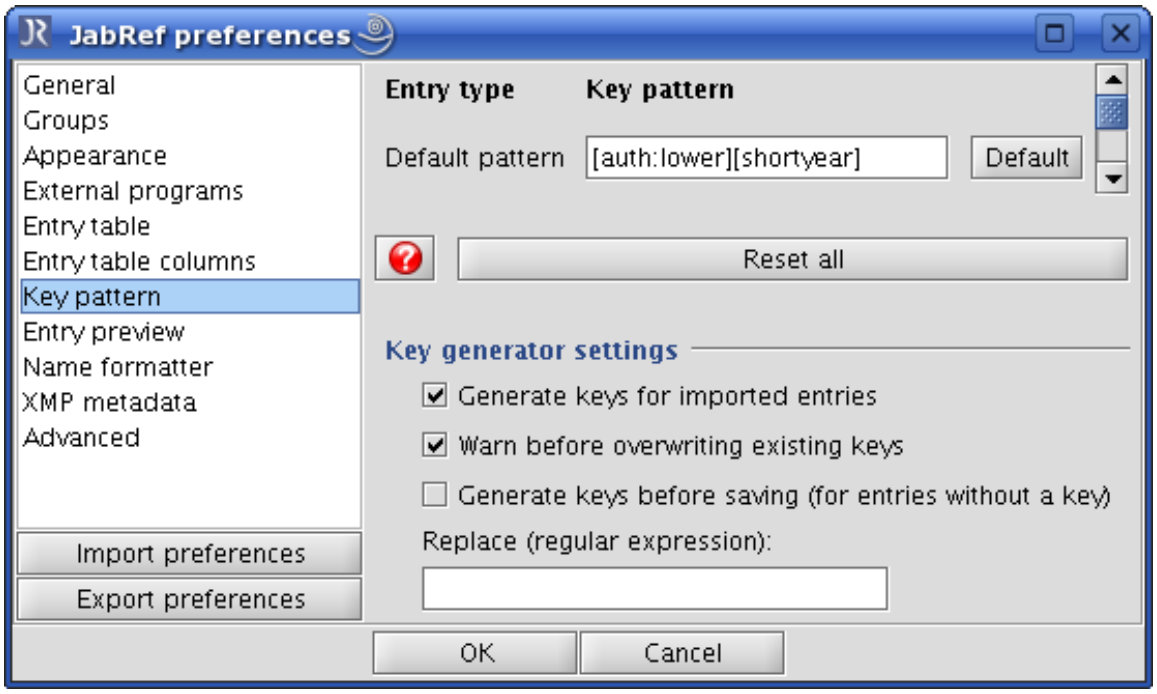

Figure 3: Specify default key pattern in **JabRef**

Second, we add the BibTeX field `location` for information about the location, where the publication is available as hard copy (e.g. a book or a copy of an article). This field can contain the name of the user who has the hard copy and where he has it or the name of a library and the shelf-mark. This field can be added in **JabRef** by selecting `Options` → `Set up general fields` and adding the word `location` (using the semicolon (`;`) as delimiter) somewhere in the line that starts

with `General:` (see figure 4).

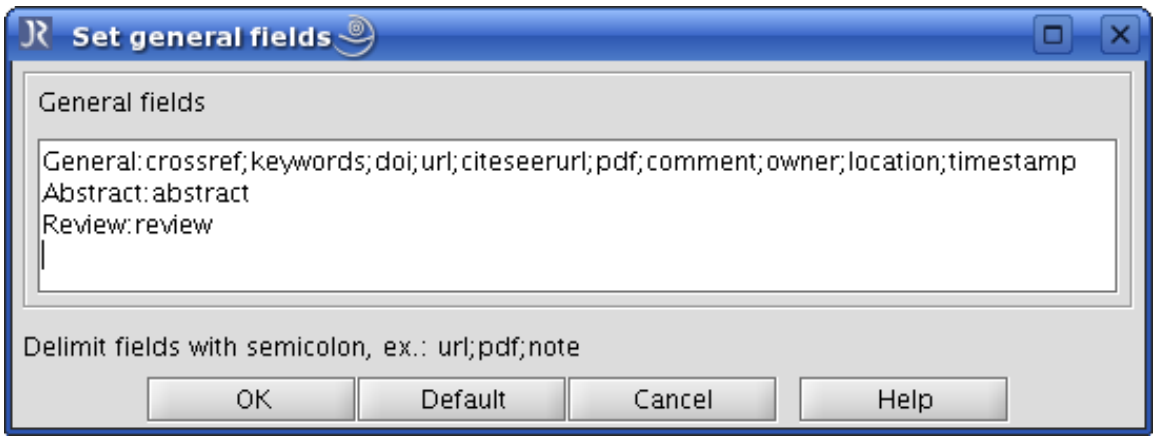

Figure 4: Set up general fields in **JabRef**

Third, we put all PDF files of publications in a specific subdirectory in our file server, where we use the BibTeX key as file name. We inform **JabRef** about this subdirectory by selecting `Options` → `Preferences` → `External programs` and adding the path of the this subdirectory in the field `Main PDF directory` (see figure 5). If a PDF file of a publication is available, the user can push the `Auto` button left of **JabRef**'s `Pdf` field to automatically add the file name of the PDF file. Now, all users who have access to the file server can open the PDF file of a publication by simply clicking on **JabRef**'s PDF icon.

If we send the LaTeX source code of a project to a journal, publisher, or somebody else who has no access to our common `texmf` tree, we do not include our entire bibliographic data base, but extract the relevant entries with the Perl script **aux2bib** [13].

7 Conclusion

This paper describes a possible way to efficiently organise the collaborative preparation of scientific LaTeX documents. The presented solution is based on the **Subversion** version control system and several other software tools and LaTeX packages. However, there are still a few issues that can be improved.

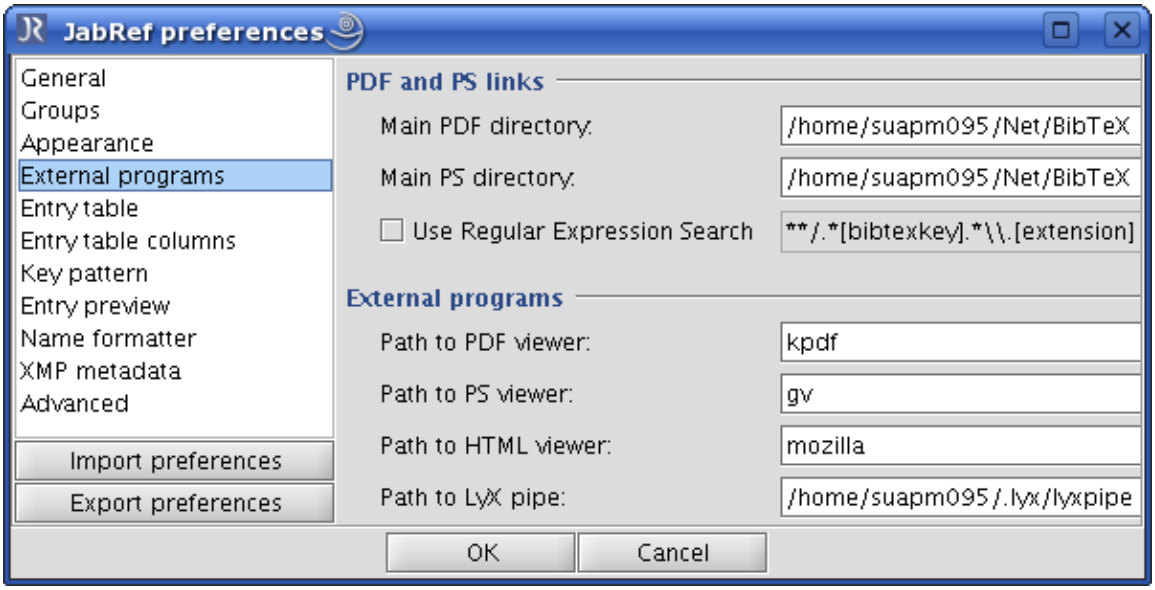

Figure 5: Specify "Main PDF directory" in **JabRef**

First, we plan that all users install the same LaTeX distribution. As the "TeX Live" distribution [28] is available both for Unix *and* MS Windows operating systems, we might recommend our users to switch to this LaTeX distribution in the future.[12]

Second, we consider to simplify the solution for a common bibliographic data base. Currently it is based on the version control system **Subversion**, the graphical BibTeX editor **JabRef**, and a file server for the PDF files of publications in the data base. The usage of three different tools for one task is rather challenging for infrequent users and users that are not familiar with these tools. Furthermore, the file server can be only accessed by local users. Therefore, we consider to implement an integrated server solution like **WIKINDX** [30], **Aigaion** [22], or **refBASE** [25]. Using this solution only requires a computer with internet access and a web browser, which makes the usage of our data base considerably easier for infrequent users. Moreover, the stored PDF files are available not only

12. Currently, our users have different LaTeX distributions that provide a different selection of LaTeX packages and different versions of some packages. We solve this problem by providing some packages on our common `texmf` tree.

from within the department, but throughout the world.[13] Even Non-LaTeX users of our department might benefit from a server-based solution, because it should be easier to use this bibliographic data base in (other) word processing software packages, because these servers provide the data not only in BibTeX format, but also in other formats.

Based on this paper, I have created a "Wikibook" on this subject [31]. All readers are encouraged to contribute to this book by adding further hints or ideas or by providing further solutions to the problem of collaborative writing of LaTeX documents.

Acknowledgements

I thank Francisco Reinaldo and Géraldine Henningsen for comments and suggestions that helped me to improve and clarify this paper, Karsten Heymann for many hints and advices regarding LaTeX and **Subversion**, and Christian Henning as well as my colleagues for supporting my intention to establish LaTeX and **Subversion** at our department.

References

[1] Mark James Adams. wdiff wrapper for svn. http://textsnippets.com/posts/show/1033.

[2] Achim D. Brucker. LaTeX package 'svninfo'. http://www.ctan.org/tex-archive/macros/latex/contrib/svninfo/.

[3] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion*. O'Reilly Media, 2007. http://svnbook.red-bean.com/.

[4] Patrick W. Daly. LaTeX package 'custom-bib'. http://www.ctan.org/tex-archive/macros/latex/contrib/custom-bib/.

13. Depending on the copy rights of the stored PDF files, the access to the server — or least the access to the PDF files — has to be restricted to members of the department.

[5] Aidan Delaney. Writing academic papers using Latex and Subversion (part 1). http://blogs.linux.ie/balor/2007/05/23/ [accessed 18-July-2007], May 2007.

[6] Dynamic Network Services, Inc. DynDNS – dynamic DNS service. http://www.dyndns.com/.

[7] Joachim Eibl. KDiff3. http://kdiff3.sourceforge.net/.

[8] Thomas Esser. teTeX. http://www.tug.org/tetex/.

[9] Jürgen Fenn. Managing citations and your bibliography with BibTeX. *The PracTEX Journal*, 4, 2006. http://www.tug.org/pracjourn/2006-4/fenn/.

[10] Free Software Foundation. wdiff. http://www.gnu.org/software/wdiff/.

[11] Gertjan P. Halkes. dwdiff. http://os.ghalkes.nl/dwdiff.html.

[12] Jean-Olivier Irisson. LaTeX bibliography styles database. http://jo.irisson.free.fr/bstdatabase/.

[13] Vivek Khera. BibTeX tool 'aux2bib'. http://www.ctan.org/tex-archive/biblio/bibtex/utils/bibtools/aux2bib, 1992.

[14] Kile Team. Kile – an integrated LaTeX environment. http://kile.sourceforge.net/.

[15] Igor V. Kovalenko and Julien Dumont. eSvn – a cross-platform GUI frontend for the Subversion revision system. http://zoneit.free.fr/esvn/.

[16] Richard Lewis. LaTeX package 'svn'. http://www.ctan.org/tex-archive/macros/latex/contrib/svn/.

[17] Nicolas Markey. Tame the BeaST. the B to X of BibTeX. http://www.ctan.org/tex-archive/info/bibtex/tamethebeast/ttb_en.pdf, 2005. Version 1.3.

[18] Oren Patashnik. Designing BibTeX styles. http://www.ctan.org/tex-archive/info/biblio/bibtex/contrib/doc/btxhak.pdf [accessed 18-July-2007], February 1988.

[19] Martin Scharrer. LaTeX package svn-multi. http://www.ctan.org/tex-archive/macros/latex/contrib/svn-multi.

[20] Christian Schenk. MiKTeX. http://www.miktex.org.

[21] TeX Users Group. A directory structure for TeX files. http://www.tug.org/tds/tds.html [accessed 18-July-2007], 2004. version 1.1.

[22] The Aigaion Developers. Aigaion – a web based bibliography management system. http://www.aigaion.nl/.

[23] The CTAN team. CTAN – The Comprehensive TeX Archive Network. http://www.ctan.org.

[24] The JabRef Developers. JabRef – an open source bibliography reference manager. http://jabref.sourceforge.net/.

[25] The refBASE Developers. refBASE – web reference database. http://refbase.sourceforge.net/.

[26] The Subversion developers. Subversion. http://subversion.tigris.org/.

[27] The Subversion developers. Subversion: Clients and plugins. http://subversion.tigris.org/links.html.

[28] The TeX Live Developers. TeX Live. http://www.tug.org/texlive/.

[29] The TortoiseSVN developers. TortoiseSVN – a subversion client implemented as a windows shell extension. http://tortoisesvn.tigris.org/.

[30] The WIKINDX Developers. WIKINDX. http://wikindx.sourceforge.net/.

[31] Wikibooks. Collaborative writing of LaTeX documents. http://en.wikibooks.org/wiki/LaTeX/Collaborative_Writing_of_LaTeX_Documents.

[32] Wikipedia. List of revision control software. http://en.wikipedia.org/w/index.php?title=List_of_revision_control_software&oldid=145420234 [accessed 19-July-2007], 2007.