The PracTEX Journal, 2007, No. 2
Article revision 2007/05/25

The PracTEX Journal: Making an Electronic Journal with web tools, Wiki and version control

Paul A. Blaga*

Address "Babeş-Bolyai" University of Cluj Napoca,
Faculty of Mathematics and Computer Science
1, Kogălniceanu Street,
400609 Cluj Napoca,
Romania

Abstract This paper describes how an issue of The PracTEX Journal is made up, and focuses on the software tools used. The tools used are the wiki, the *subversion* version control system, and Perl scripts. One of the goals of The PracTEX Journal, and the reason for using these tools, is to create an environment where a small team of volunteers can put together an online journal with a minimum of time and work.

1 Introduction

The PracTEX Journal is an international online journal aiming to help the spread and the development of different variants of TEX, from the "classical" Plain TEX and LATEX to "newcomers" such as ConTEXt. The articles published by The PracTEX Journal range from expository papers to technical ones, but all share a practical character.

There are some interesting aspects of the journal we would like to mention. First of all, it is published exclusively in electronic format. Second, the journal is compiled by editors from all around the world (Europe, United States, Australia, India) working together and simultaneously with the manuscripts. This is made possible by using two modern tools of electronic publications: the *wiki* and a version control system, *subversion*. The wiki is used by the editors to coordinate the

*The paper was actually written with the cooperation of Lance Carnes and Dave Walden. Thanks, also, to Francesco Reinaldo and Yuri Robbers for suggestions.

assembly of each issue. It can also be added to by readers and others who want to suggest an article idea. For us, it is an invaluable opportunity to exchange ideas, and a way of tracking the progress of producing an issue. The second tool, the *subversion* version control system, allows the editors to work simultaneously on the same manuscripts. Editors can update their local files to include all changes by other editors, so that at any point in time each editor has a current set of files for the current issue. It also lets the editors add or change files and has safeguards to prevent conflicting updates.

We shall explain, first, how these two tools work and then we will provide a description of how the journal itself is made up.

2 The Editorial process

The machinery of The PracTEX Journal has three main components: an automatically generated web site (designed by Dave Walden), a wiki (hosted by pctex.com), and the version control system (originally hosted at pctex.com and now at tug.org.)

Each issue of The PracTEX Journal has an issue editor who chooses a theme for the issue and solicits "theme" and other articles.

Now the wiki enters the stage. This is the place where prospective authors announce their projects. They can either edit the wiki page directly and add the title of their articles, or announce them to the editorial staff in order to have them take care of the process. Once an article is chosen for publication, a production editor is assigned to make it ready for publication. The *production editor* works with the author during the review and proofreading process. The progress of all articles is tracked in the wiki http://wiki.pctex.com/index.php?title=PracTeXJournal. This way the entire editorial staff is being kept up to date on the production process.

3 The PracTeX Journal wiki

The PracTEX Journal wiki is a web site that can be edited by the production editors in order to track the progress of the current issue. See the wiki at http://wiki.pctex.com/index.php?title=PracTeXJournal. A wiki is like an electronic white

board, but instead of using felt-tip pens it is edited online (much cleaner, and no smelly pens!). We note articles that have been received, what state they are in (proposed, received, in review, final), who is responsible for them, and other information.

For example, here is the wiki entry (in wiki syntax) for the article you are now reading.

```
|PracTeX Journal: Making an Electronic Journal with Wiki
   and Subversion || Paul Blaga et al || Paul || Proposed  || ~~~~~
```

This shows the working title, author, editor in charge, status (Proposed), and the date this was entered in the wiki. (The wiki changes `~~~~~` into the current date and time.)

As the issue progresses the schedule is updated. You are welcome to look at the wiki and see the progress of future issues. While you are there feel free to add an article idea http://wiki.pctex.com/index.php/PracTeXJournal#Article_ideas.

3.1 More about Wiki's

The name "wiki" is, surely, familiar to any web navigator, if only because of the free web encyclopedia Wikipedia http://wikipedia.org, probably one of the richest online encyclopedias. It is a good example of the wiki philosophy, because it was written by its readers and is constantly being updated.

To quote the beginning of the chapter on wikis from one of the latest books on virtual team management ([2]), "Wikis are collaborative websites that allow users to add and edit content." Behind these friendly websites there is software supporting the functions of the wiki. Usually, the word "wiki" is used both for the web page itself and for the software. The wiki was invented in 1994 by Ward Cunningham, and he also used this name for the first time. He didn't actually *invent* the name, it's just a Hawaiian word meaning "fast" or "quick". You can find the whole story in many places on the web (and, of course in the Wikipedia, as expected).

There are several wiki engines; some are free, some are commercial. We have no intention whatsoever to describe them. The interested reader can have a look

at the very nice book [4], one of the very few (if not the only!) entirely dedicated to wikis.

The way a wiki works is not difficult to understand. First of all, it is not entirely "democratic", as one might deduce from what we said so far. Any wiki page is created and put on the web by an *administrator*, which always has more rights and can restrict some of the rights of the users, if he thinks it's appropriate to do so.

No matter what software is used, on most wiki pages there are available at the least the following operations (beside the editing itself):

registration – usually, the reader is asked to register and then login before making any modification to the web page. Sometimes, the registration requires the approval of the administrator.

history – this is one of the most important features of a wiki website. It simply lets you track the previous versions and, if necessary, you can revert to any of these versions. This is essential both because you can correct errors made by changing involuntarily the contents of the site.

locking pages – the administrator can lock some of the page and only him will be able to modify these pages.

search – which has the obvious meaning, although not all the wikis would allow performing the entire range of searches.

recent changes – indicates what pages from the site were modified most recently.

As concerns the editing process, we have to say that so far there is no general consensus on the editing commands; each wiki software comes equipped with its own set. Most wikis allow only some elementary text modifier elements, some of the newer ones, however, are a lot more complex.

4 Version control for editing

While making up the first issue of The PracTEX Journal there was one person who coordinated all the article files. When an editor updated an article he would email it to the person in charge of files who would then update it into the issue archive. This person had to keep track of which files needed to be added or

updated, and hopefully put it them the right place. With ten or twelve articles, each with several revisions, this was fast becoming a nightmare. By the time the first issue was published the person in charge of files let it be known that this was a horrible, miserable job, and there had to be a better way.

We hit on a better way. It's called a version control system (VCS), a software system for maintaining versions of a large set of files. Since some of the editors are also software engineers, this was a natural choice; a VCS is used on most all software projects, including the TEX User Group's TEX Live project.

The VCS is the electronic form of the person in charge of files. It is a database of files and resides on a computer that can be accessed via the internet. Using some fairly intuitive commands the user can tell the VCS to "update this file into the TPJ 2007-2 issue" or "give me the latest version of the file `team.tex`", and other tasks. All files are sent and received automatically via the internet.

For those editors who had not used a VCS before it was a bit of a challenge to learn, but after a short while it paid off. We chose a VCS called *subversion*, a popular free software system. The usual way of using *subversion* is with a command line (see the next section for some examples). There are also some nice graphical user interfaces (GUI) that can be used with Windows, Linux and other systems. The editors who are not software developers (and those who are too lazy to memorize all the command line options) use a GUI interface.

Briefly, here is how a production editor would use *subversion* to work on the file for this article `team.tex`. (`svn` is the command to run *subversion*.)

- Starting up. Before editing, the local file copies must be brought up to date. This step updates any work was done by other editors into the local copy. Just type

  ```
  svn update
  ```

- Editing. Once the local files are current, the editor may use his/her favorite text editor to make changes to the files. Just open `team.tex` in a text editor and work as usual.

- Winding up. Once the edits are finished it is time to put the updated file in the archive, so that the other editors will be able to benefit from (or complain about) the changes made. To put the changes into the master archive, type

  ```
  svn commit -m "corrected a bunch of errors - can't you guys spel?"
  ```

The text after `-m` is a comment that will appear in the archive log.

That's a typical *subversion* session. It can do dozens of other useful things as well — see the next section for more details.

4.1 *Subversion* and version control systems

Wikis are, first all, great ways of *exchanging information*. Version control systems ([3, 1]), are a powerful tool for online *collaboration*. These systems are used mostly by programmers and others to coordinate a project composed of a set of interrelated files. The version control system used by the PracTEX Journal is *subversion*.

Subversion, and other version control systems, are at the core a database consisting of all the files used in a project. To access and update this database there are several user interfaces available for most all operating systems. Some of them have a graphical user interface (GUI), while other are operated by command line. We shall describe the command line version, which is used in Unix and similar systems.

Subversion allows several people to work simultaneously on the same project. The project's subversion database is located on a system which can be accessed from the internet. Collaborators are given an account which allows them to access it and modify it.

Thus, the first step in working with subversion is the creation of a space for the project. This step is performed by an *administrator* and the corresponding database (and the contents of the project which is stored there) is called a *repository*. Then the administrator creates accounts for the users who are authorized to work on the project.

At this point the users enter the stage. We assume that the interface software is already installed. The first step is to create copies of the project files on their own computers. This is very easy to do (once they are logged on). First, they have to create a directory where everything is supposed to lie (for instance, a "practex" directory, on the drive D). Then change directory to the chosen one (this is the windows version, of course, the modifications for Unix should be obvious):

```
cd D:\practex
```

Now, the user can download the repository, to create a local copy. For instance, for PracTEX the command is:

```
svn co svn://tug.org/pracjourn/trunk
```

This may take a while depending on the dimension of the project and the speed of the connection. The good news is that the steps so far have to be done only once.

Now, that everything is on your computer, you can modify everything you want and then propagate the changes to the central repository. Typically, a work session would look like that:

First of all, copy from the central repository the latest changes. This is done with the command:

```
svn update
```

All the changes will be downloaded. You will also get a list of the changed files/directories and the number of the version of the project.

Now you can do your everyday work. Normally, you will perform three kind of operations: editing files, creating files and creating directories. Beware that even if you create by the commands specific to your operating system, files and directory, or you copy them on your local repository from somewhere, they don't exist for subversion and will not be uploaded to the central repository unless you let subversion know what you did. This is done by the subversion command `add`:

```
svn add file1 file2 directory1
```

If you're not sure whether you "added" all the files and directory you intended to do, or did not modify all the necessary files, you can run a check:

```
svn status
```

and you will get the list of the modifications you made.

And now, the final step, the commitment (which is important in subversion, as well):

```
svn commit -m "Describe the changes you made."
```

the option `-m` stands, of course, for "message".

Now all your changes are propagated to the central repository. Notice that no changes is propagated until you give the commit command.

An important pair of commands allow the locking and unlocking of a file/directory. If you want to modify a file and you want to be sure that nobody will perform another modification at the same time with you, you can use the command:

```
svn lock file
```

and now nobody will be able to modify the file until you commit your changes or until you unlock the file with the command

```
svn unlock file
```

5 Putting everything together

After the review process, and an article is being worked on for the current issue, a directory is created and all files related to the article are added. This makes the paper available to the entire staff, via the VCS, and allows the production editor to make simple changes or fix problems in conjunction with the author, or even to replace the entire paper by an updated version submitted by the author. In the final stages the production editor changes the article header to reflect that this is the accepted version rather than a submitted paper, and updates the date of acceptance. Then an abstract and a short author biography for the journal website is entered. This will be used during the automated website generation.

When all papers are ready, the editorial has been written, etc., the editor in charge will take care of the final touches, which includes producing a single pdf file of the entire journal, and the current issue will be placed on a testing web site. The editorial staff as well as a group of carefully selected proofreaders then check for any remaining problems. After a short testing period, the "final" web site for the issue is made public.

6 Web site generation and updating

The *TPJ* website is generated by a Perl script. There are at least three advantages in having a website generation program such as we use for *TPJ*: (1) the HTML pages for the tables of contents for all issues and for each piece in each issue have consistent formats; (2) if we decide on a change in format, we can change every piece and every issue with (hopefully) a small change to the program or (better yet) an HTML template for a particular format; (3) as the program is processing all of the pieces in every issue, it also makes title, author, and `bibtex` indexes across all issues and pieces.

6.1 More about site generation

For readers who are interested, the program works as follows: The svn repository for the journal contains a directory for each issue, with directory names that give the issue number, e.g., `pracjourn/2007-2/`. Each such directory includes a "driver file" that tells the generation program the titles and authors of each piece in the issue as well as grouping the pieces into the categories of Notices, Articles, and Columns. The HTML table of contents for each issue is generated from the data in this driver file. There is also a directory for each piece in the issue. Various of the following are included in this directory: an abstract of the piece, a PDF of the piece, HTML for the piece in the instance where there is no PDF for the piece, etc. The website generation program processes the information in the driver file for each issue, creating the HTML page for the table of contents for the issue and the HTML pages for each piece in the issue, as well as the various indexes. When we are producing a new issue, we typically run the program in a mode where only the HTML pages for the latest issue are generated along with the indexes for all issues. The program also uses a couple of other small driver files to create the left column of the home page and some HTML pages linked to from the left column.

7 Application to other journals

It's possible that some of the techniques described here could be applied to other print or electronic journals. The wiki is essentially an electronic white board and can be learned easily. Whenever someone involved with a project has a new idea or a new method, he or she can post it to the wiki. It's a great way to keep everyone on the same page (publishing pun).

If a journal, either print, electronic or both, has a number of editors who are updating articles regularly, the Version Control System (VCS) is a useful tool. With a graphical interface it's easy to use. It avoids collisions (editor A accidentally overwriting editor B's work), it provides a log of everything that has been done, it allows an earlier version of file to be restored, and many more functions. It will even show the differences between versions of a file — if editor A takes the day off and her colleague editor B fills in for her, when she returns the VCS will quickly show all changes made.

There are a number of other little tricks we have employed to make The PracTEX Journal a low-cost way for a volunteer group to publish a quarterly journal. If you would like more information on any of the techniques discussed feel free to get in touch mailto:pracjourn@tug.org. You may also register at the wiki site http://wiki.pctex.com/index.php?title=PracTeXJournal and post your comments.

References

[1] Berlin, D., Rooney, G.: *Practical Subversion*, 2nd edition, Apress, 2006

[2] Brown, M.K., Huettner, B., James-Tanny, C.: *Managing Virtual Teams: Getting the Most from Wikis, Blogs and Other Collaborative Tools*, Wordware Publishing, Inc., 2007

[3] Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M.: *Version Control with Subversion*, O'Reilly, 2004

[4] Ebersbach, A., Glaser, M., Heigl, R.: *Wiki Web Collaboration*, Springer, 2006