

TALUG Meeting Notes

March 29, 2008

Subversion: Presented by Ian Wilson

[Collabnet](#) is the industry's most widely used collaborative platform for software development. They support the development of [Subversion](#). Ian is an RHCE certified engineer who has worked for Collabnet for two years.

The meeting notes are a general overview of the presentation, demo, and discussion that occurred at the meeting. For more detailed information, see the link to Ian's presentation above.

Ian had arranged for a number of copies of the [Subversion book](#), but they didn't arrive in time. They will be available at the next meeting.

Source Code Management

Source code management (SCM) tools provide a way to maintain an up-to-date master copy of code, which can be kept in a central location where users can check out files, edit them and check them back in. There are a number of source code management programs that have evolved over the years, a partial chronological list of important developments is as follows:

1. The VMS File System – The file system of the VMS operating system maintained revisions on local disk.
2. [SCCS](#) – First source code revision control system, developed by Bell labs.
3. [RCS](#) – Successor to SCCS, no network support.
4. [CVS](#) – First SCM with client \iff server architecture.

The next step in the evolution of SCM tools was Subversion, tagged as “CVS done right.” Currently, there are a number of choices for a source code management system, some of the most popular choices are as follows:

- [Subversion](#)
- [Git](#)
- [Bazaar](#)
- [Mercurial](#)
- [Darcs](#)
- [Monotone](#)

One of the problems with current SCM tools is that with a small repository and lots of developers, you will encounter lots of locks. Currently, this is a problem with both CVS and Subversion. Git has a way around this, where each developer has their own branch, and then code can be merged.

It was noted that source code management tools can be used locally on a single computer, but the true power is in network deployment with many users.

Demo

There was an interactive demo where Ian set up a [Source Forge host](#) that we could log into to create a project. We went through the following steps (at a fairly rapid pace):

- From the home tab in the source forge website, we clicked the *Projects* link, and the *Create Project* link.
- Clicked on the *Project Home* link, and then on the *Source Code* link.
- Used the provided SVN checkout command:
`svn checkout --username <insert name> http://sfee-hosted.com/svn/repos/<insert repo>`
This checked out a copy of the project, which initially contained nothing. On your local machine, it creates the directory for the project, and the `.svn` directory which contains all the project info so that you no longer have to specify usernames, passwords, or url's to check in or check out code.
- Updated our local copy with `svn update`.
- Added a file, and then committed back to the server with `svn commit`. When committing code, you should submit with comments so that others know what you were working on.
- Modified the file again, performed an `svn diff` and then committed back to the repository.
- Went back to the source forge web-gui and looked through the revision history. Revision history shows line by line changes, and color codes the revision levels.
- With very few changes, the revision history didn't show us much. In order to showcase the more powerful features, we browsed through some active projects.

Note: Subversion has multiple “filesystems” or “backends” that can be used, including [FSFS](#) and [BDB](#).

Subversion Basics

See the presentation for better explanation of basic use of Subversion. Some of the basic commands used are shown below:

- `svn checkout --username <username> <http://address/of/repo/>` – Get an up-to-date copy of the project.
- `svn revision <file>` – Displays what version a file is on.
- `svn add <file>` – Add a new file.
- `svn revert <file>` – Reverts to master copy.
- `svn checkin <file>` – Upload file to repository.
- `svn diff -r3:4 <file>` – Find differences between revisions 3 and 4 of the file specified.
- `svn update` – Sync files with repository.
- `svn commit` – Upload all changes to repository.
- `svn merge -r5:6 http://path/to/branch` – Merge revisions 5 and 6. Be careful with this one.
- `svn resolve` – Fix conflicts with files.

Sometimes to new users the commands (Check in, check out, merge, diff, update) may sound complicated. As a result, sometimes developers rely heavily on their local copies because it's easier. But this is bad practice, as it impedes the development of others.

It was noted to check in (commit) and update regularly. This should be done several times a day on active projects, so that others can see what you've been working on.

Questions and Comments

- File locking can become a problem for files that are frequently edited by multiple users. If you are going to do lots of edits, you can lock a file to prevent others from using it until you are finished. Both Subversion and CVS work in this fashion. The Linux kernel development group currently uses Git, which minimizes these problems in large projects with lots of updates.
- We looked at a project that uses Git. Each developer can have their own master copy, and can merge changes. Ian thinks that Git is maybe the way to go (from open source perspective), but Git probably won't get much traction in corporate world.
- Git is actually a cool program, with some useful features. Git has backward compatibility, so if you use Git on a project that used to use CVS, Git will act like CVS and no migration is required. People can still use features that they are familiar with. One downside (for some users) is the lack of Windows support.

Q: Due to difficulties some users have with merging, some people convert to Git, merge, and then convert back to Subversion. What is the best way to merge in Subversion?

A: Currently, this might be your best solution if you need to do a lot of merging.

Q: What if you use your Subversion server as a local computer? Can use use the repository directly? Are there problems accessing the repository?

A: You can still use repositories even if you are on the local machine. The syntax would be:

```
svn checkout svn://path/to/repository
```

You should always use repositories (and update often). You cannot directly edit the repository, as it is stored in a database format.

- Source code management can't fix lack of communication between developers. Subversion will not allow commits for the same file from more than one user. It will give an error and ask which file to use. Any source control management program is just a tool to help you develop software. It can't make up for poor development practices.
- Pre and post commit hooks were discussed. These are scripts that automatically run either before, or after committing an update. There is a lot of power in using these hooks.

Q: Can you permanently delete a file and all its history from the repository?

A: No. This is because this software is used in the corporate world, and history (accountability) is necessary. The file will still be in that "revision" for accountability purposes, however, you can delete that file from future revisions using the `svn delete` option. You *can* permanently delete files if you're using `mod_dav_svn`, however, some of the others won't.

- Do not check binary files into Subversion, it has the potential to break things, especially with a lot of users.

Q: How does Subversion compress?

A: Subversion repositories compress really well on disk. Subversion uses an internal compression format that has just a shade better performance than `gzip`, while Git uses `zlib` which means Git can compress even further (10 times better than `gzip` in some cases).

Q: What are the top three version control systems?

A: 1. Subversion
2. Git
3. Bazaar

- [Git](#) works well with large projects with lots of commits (changes). However, Git provides no central user management, and no access control. Most companies need these features (for audit-ability, customer delivery, and for legal considerations). As a result, it is limited in its commercial application. It works well for open source projects, and is used for Linux kernel development.
- [Bazaar](#) is source management tool of Launchpad, developed by Canonical. It is written in python, and is a serious contender for the corporate environment. It was developed with a focus on usability and simplicity (it *Just Works*).

Q: What are the web-gui options for Subversion?

A: [Sourceforge Enterprise Edition](#), [Fisheye](#), [websvn](#), and more. A more complete list of available options can be found on the [tigris.org](#) site. Sourceforge is the most powerful of the options, and offers a number of cool features including burndown charts for development monitoring. However, for most users the websvn option is sufficient.

- One of the potential drawbacks of Subversion is a lack of [horizontal scalability](#). As a result, if a server goes down users will no longer be able to access the repository until the server comes back up.
- [Cruise control](#) is a framework for continuous build development that works with CVS. It compiles and checks the code as part of the check in. It is especially useful for developers in more than one location. Ian noted that we are getting to a point in software development where continuous build and software life-cycle management are necessary.

Links

Featured

[SVN Book](#)

[Google TechTalk Git](#)

[Google TechTalk Linus Trovalds On Git](#) — Bashes CVS and SVN developers and users.

[Git User Manual](#)

Subversion Specific

[Google Code University: Software Configuration Management](#)

[A Visual Guide To Version Control](#)

[Setting Up Subversion And Websvn On Debian](#)

Git Specific

[Git SVN Workflow](#)

[Git Guide](#)

[Git Magic](#)

Election Results

At the end of the meeting, the results of the election were announced. The new officers are as follows:

President: Neal Dudley

Vice-President: Denny Pettee

Secretary: Scott Vargovich

Treasurer: Steve Tryc

After Meeting Activities

The presentation section of the meeting finished up around 4:00, and a good number of people stayed around to talk about Linux in general, and troubleshoot a few computers. Among the topics discussed (that I overheard anyway) were the future direction of TALUG, under the leadership of the new officers, and of course food.

After the meeting wrapped up, a group of TALUG members went to eat at Ipoh, a local Chinese restaurant. It was noted that we eat a lot of Chinese food, so after the next meeting we will choose something different.