

Comparison of CVS and Subversion

D.J. Worth and C. Greenough

October 2005

Abstract

Source code management is one of the basics of good software development. If used properly it provides a trail of the changes made to the source code and documentation, as well as being an aid when bug-fixing. In this document we compare two of the major source code management technologies - CVS and Subversion - outlining their similarities and differences under the following headings

- repository operations
- features
- technical status
- user interfaces
- licenses

We conclude with a list of other source code management systems and a description of setting up CVS and Subversion on the Software Engineering Group computer system.

Keywords: source code management, CVS, Subversion

Email: d.j.worth@rl.ac.uk or c.greenough@rl.ac.uk
Reports can be obtained from www.softeng.cse.clrc.ac.uk

Software Engineering Group
Computational Science & Engineering Department
Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire OX11 0QX

© **Council for the Central Laboratory of the Research Councils**

Enquires about the copyright, reproduction and requests for additional copies of this report should be address to:

Library and Information Services
CLRC Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire OX11 0QX
Tel: +44 (0)1235 445384
Fax: +44 (0)1235 446403
Email:library@rl.ac.uk

CLRC reports are available online at:

<http://www.clrc.ac.uk/Activity/ACTIVITY=Publications;SECTION=225;>

ISSN 1358-6254

Neither the Council nor the Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations

Contents

1	Introduction	1
1.1	CVS	1
1.2	Subversion	2
2	Repository Operations	3
2.1	Atomic Commits	3
2.2	Files and Directories Moves or Renames	3
2.3	File and Directories Copies	3
2.4	Remote Repository Replication	3
2.5	Propagating Changes to Parent Repositories	3
2.6	Repository Permissions	4
2.7	Changesets Support	4
2.8	Tracking Line-wise File History	4
3	Features	4
3.1	Ability to Work Only on One Directory of the Repository	4
3.2	Tracking Uncommitted Changes	4
3.3	Per-file Commit Messages	4
4	Technical Status	5
4.1	Documentation	5
4.2	Ease of Deployment	5
4.3	Command Set	5
4.4	Networking Support	5
4.5	Portability	6
5	User Interfaces	6
5.1	Web Interface	6
5.2	Availability of Graphical User Interfaces	6
6	License	6
7	Summary	6
A	Other SCM Systems	7
A.1	SCCS	7
A.2	RVS	7
A.3	Aegis	8
A.4	Perforce	8
A.5	Visual SourceSafe	8
B	CVS Set Up	9
C	Subversion Set Up	10
D	Converting CVS Repositories to Subversion	12

1 Introduction

This note provides a comparison of two source code management (SCM) tools:

- CVS (Concurrent Versions System) - <http://www.nongnu.org/cvs/>
- Subversion - <http://subversion.tigris.org/>

The information was found at [1] and considers the following aspects of the tools:

- Repository operations
- Features
- Technical status
- User interface
- License

We will give a brief overview of each system here to set the context of what follows in the next sections.

1.1 CVS

- Client/server CVS enables disparate developers to function as a single team. The version history is stored on a single central server and the client machines have a copy of all the files that the developers are working on. Therefore, the network between the client and the server must be up to perform CVS operations (such as checkins or updates) but need not be up to edit or manipulate the current versions of the files. Clients can perform all the same operations which are available locally.
- CVS' basic version control functionality maintains a history of all changes made to each directory tree it manages, operating on entire directory trees, not just single files.
- CVS supports branches allowing several lines of development to occur in parallel and providing mechanisms for merging branches back together when desired.
- CVS can tag the state of the directory tree at a given point, recreate that state and display the differences between tags or revisions in the standard diff formats.
- CVS runs scripts which you supply to log operations or enforce site-specific policies.
- CVS has Unreserved Checkouts allowing more than one developer to work on the same files at the same time.
- CVS provides a flexible modules database that provides a symbolic mapping of names to components of a larger software distribution. It applies names to collections of directories and files. A single command can manipulate the entire collection.
- CVS gives project managers fine control over the development process, e.g. by checking changes about to be committed, checking log messages or e-mailing other developers about changes.
- Remote operation is efficient, transmitting only those files which have changed. When appropriate, CVS transmits patches to files and verifies the results rather than sending entire files. CVS can compress the text it transmits.
- Remote operation is authenticated. CVS can use the industry standard Kerberos protocols to verify the identity of the remote user. Kerberos is much more secure than the source-address authentication provided by the ordinary rlogin and rsh protocols. CVS can also work with ssh, a secure replacement for rsh, or use straight password authentication.

1.2 Subversion

Subversion is meant to be a better CVS, so it has most of CVS' features. Generally, Subversion's interface to a particular feature is similar to CVS', except where there's a compelling reason to do otherwise.

- Directories, renames, and file meta-data are versioned. Lack of these features is one of the most common complaints against CVS. Subversion versions not only file contents and file existence, but also directories, copies, and renames. It also allows arbitrary metadata ('properties') to be versioned along with any file or directory, and provides a mechanism for versioning the 'execute' permission flag on files.
- Commits are truly atomic. No part of a commit takes effect until the entire commit has succeeded. Revision numbers are per-commit, not per-file; log messages are attached to the revision, not stored redundantly as in CVS.
- Apache network server option, with WebDAV/DeltaV protocol. Subversion can use the HTTP-based WebDAV/DeltaV protocol for network communications, and the Apache web server to provide repository-side network service. This gives Subversion an advantage over CVS in interoperability, and provides various key features for free: authentication, path-based authorization, wire compression, and basic repository browsing.
- Standalone server option. Subversion also offers a standalone server option using a custom protocol (not everyone wants to run Apache 2.x). The standalone server can run as an inetd service, or in daemon mode, and offers basic authentication and authorization. It can also be tunnelled over ssh.
- Branching and tagging are cheap (constant time) operations. There is no reason for these operations to be expensive, so they aren't. Branches and tags are both implemented in terms of an underlying 'copy' operation. A copy takes up a small, constant amount of space. Any copy is a tag; and if you start committing on a copy, then it's a branch as well. (This does away with CVS' 'branch-point tagging', by removing the distinction that made branch-point tags necessary in the first place.)
- Natively client/server, layered library design. Subversion is designed to be client/server from the beginning; thus avoiding some of the maintenance problems which have plagued CVS. The code is structured as a set of modules with well-defined interfaces, designed to be called by other applications.
- Client/server protocol sends diffs in both directions. The network protocol uses bandwidth efficiently by transmitting diffs in both directions whenever possible (CVS sends diffs from server to client, but not client to server).
- Costs are proportional to change size, not data size. In general, the time required for a Subversion operation is proportional to the size of the changes resulting from that operation, not to the absolute size of the project in which the changes are taking place. This is a property of the Subversion repository model.
- Choice of database or plain-file repository implementations. Repositories can be created with either an embedded database back-end (BerkeleyDB) or with normal flat-file back-end, which uses a custom format.
- Versioning of symbolic links. Unix users can place symbolic links under version control. The links are recreated in Unix working copies, but not in win32 working copies.
- Efficient handling of binary files. Subversion is equally efficient on binary as on text files, because it uses a binary diffing algorithm to transmit and store successive revisions.

- Parseable output. All output of the Subversion command-line client is carefully designed to be both human readable and automatically parseable; scriptability is a high priority.

The following sections give a range of comparisons between CVS and Subversion.

2 Repository Operations

2.1 Atomic Commits

Support for atomic commits means that if an operation on the repository is interrupted in the middle, the repository will not be left in an inconsistent state. Are the check-in operations atomic? Are the check-out operations atomic, or can interrupting an operation leave the repository in an intermediate state?

CVS	No. Commits are not atomic.
Subversion	Commits are atomic

2.2 Files and Directories Moves or Renames

Does the system support moving a file or directory to a different location while still retaining the history of the file?

CVS	No. Renames are not supported and a manual one may break history in two.
Subversion	Yes. Renames are supported.

2.3 File and Directories Copies

Does the version control system supports copying files or directories to a different location at the repository level, while retaining the history?

CVS	No. copies are not supported.
Subversion	Yes. It's a very cheap operation ($O(1)$) that is also utilized for branching.

2.4 Remote Repository Replication

Does the system support cloning a remote repository to get a functionally equivalent copy in the local system? That should be done without any special access to the remote server except for normal repository access.

CVS	No.
Subversion	Indirectly, by using the SVN::Mirror script [2] or the svn-push utility [3].

2.5 Propagating Changes to Parent Repositories

Can the system propagate changes from one repository to another?

CVS	No.
Subversion	Yes, using either the SVN::Mirror script [2] or the svn-push utility [3].

2.6 Repository Permissions

Is it possible to define permissions on access to different parts of a remote repository? Or is access open for all?

CVS	Limited. 'pre-commit hook scripts' can be used to implement various permissions systems.
Subversion	Yes. The WebDAV-based service supports defining HTTP permissions various directories of the repository.

2.7 Changesets Support

Does the repository supports changesets? Changesets are a way to group a number of modifications that are relevant to each other in one atomic package, that can be cancelled or propagated as needed.

CVS	No. Changes are file-specific.
Subversion	Partial support. There are implicit changesets that are generated on each commit.

2.8 Tracking Line-wise File History

Does the version control system have an option to track the history of the file line-by-line? I.e: for each line can it show at which revision it was most recently changed, and by whom?

CVS	Yes - cvs annotate.
Subversion	Yes - svn blame.

3 Features

3.1 Ability to Work Only on One Directory of the Repository

Can the version control system checkout only one directory of the repository? Or restrict the check-ins to only one directory?

CVS	Yes.
Subversion	Yes.

3.2 Tracking Uncommitted Changes

Does the software has an ability to track the changes in the working copy that were not yet committed to the repository?

CVS	Yes. Using cvs diff.
Subversion	Yes. Using svn diff.

3.3 Per-file Commit Messages

Does the system have a way to assign a per-file commit message to the changeset, as well as a per-changeset message?

CVS	No. Commit messages are per change.
Subversion	No. There is no such feature.

4 Technical Status

4.1 Documentation

How well is the system documented? How easy it is to get started using it?

CVS	Excellent. There are many online tutorials and resources and an online book. The command line client also provides an online comprehensive help system.
Subversion	Very good. There is a free online book and some online tutorials and resources. The book is written in DocBook/XML and so is convertible to many different formats. The command-line client also provides a good online help system that can be used as a reference.

4.2 Ease of Deployment

How easy it is to deploy the software? What are the dependencies and how can they be satisfied?

CVS	Good. Because it is the de-facto standard, CVS is available on most systems and is easy to deploy.
Subversion	A Subversion service requires installing an Apache 2 module (if one wishes to use HTTP as the underlying protocol) or its own proprietary server. The client requires only the Subversion-specific logic and the Neon WebDAV library (for HTTP). Installation of the components is quite straightforward, but will require some work, assuming Subversion does not come prepackaged for one's system.

4.3 Command Set

What is the command set? How compatible it is with the commands of CVS (the current open-source defacto standard)?

CVS	A simple command set that includes three most commonly used commands (<code>cvs commit</code> , <code>cvs update</code> and <code>cvs checkout</code>) and several others.
Subversion	A CVS-like command set which is easy to get used to for CVS users.

4.4 Networking Support

How well is the networking integration in the system? How compliant is it with existing protocols and infrastructure?

CVS	Good. CVS uses a proprietary protocol with various variations for its client/server protocol. This protocol can be tunneled over an SSH-connection to support encryption.
Subversion	Very good. The Subversion service can use either WebDAV+DeltaV (which is HTTP or HTTPS based) as its underlying protocol, or its own proprietary protocol that can be channeled over an SSH connection.

4.5 Portability

How portable is the version-control system to various operating systems, computer architectures, and other types of systems?

CVS	Good. Client works on UNIX, Windows and Mac OS. Server works on UNIXes and on Windows with a UNIX emulation layer.
Subversion	Excellent. Clients and Servers work on UNIX, Windows and Mac OS X.

5 User Interfaces

5.1 Web Interface

Does the system have a WWW-based interface that can be used to browse the tree and the various revisions of the files, perform arbitrary diffs, etc?

CVS	Yes. CVSWeb [4], ViewCVS [5], and Chora [6].
Subversion	Yes. ViewCVS [5], Chora [6], SVN::Web [7], WebSVN [8], ViewSVN [9], mod_svn_view [10], SVN::RaWeb::Light [11], SVN Browser [12] and Insurrection [13]. Aside from that, the Subversion Apache service provides a rudimentary web-interface.

5.2 Availability of Graphical User Interfaces

What is the availability of graphical user-interfaces for the system?

CVS	Very good. There are many available GUIs: CVSGui [14] (for Windows, Mac and Linux) , Cervisia [15] (for KDE), TortoiseCVS [16] (Windows Explorer plug-in).
Subversion	Very good. There are many available GUIs: RapidSVN [17] (cross-platform), TortoiseSVN [18] (Windows Explorer plug-in), JSVN [19] (Java), etc. Most of them are still under development.

6 License

What are the licensing terms for the software?

CVS	GNU GPL (open source).
Subversion	Apache/BSD-style license. (open-source).

7 Summary

We have presented a comparison of two source code management systems, CVS and Subversion, which are the two most popular among a large collection of such systems. The systems were compared in a number of categories including repository operations, features, technical status and user interfaces and overall we recommend the use of Subversion primarily because it uses atomic commits and provides versioning for directories as well as files.

To make the switch there is a utility to convert an existing CVS repository to a Subversion repository see Appendix D.

References

- [1] SCM Comparison, <http://better-scm.berlios.de/comparison/comparison.html>
- [2] SVN::Mirror, <http://search.cpan.org/~clkao/SVN-Mirror-0.64/>
- [3] SVN Push, <http://search.cpan.org/dist/SVN-Push/>
- [4] CVSWeb, <http://www.freebsd.org/projects/cvsweb.html>
- [5] ViewCVS, <http://viewcvs.sourceforge.net/>
- [6] Chora, <http://www.horde.org/chora/>
- [7] SVN::Web, <http://freshmeat.net/projects/svnweb/>
- [8] WebSVN, <http://websvn.tigris.org/>
- [9] ViewSVN, <http://viewsvn.berlios.de/>
- [10] mod_svn_view, http://www.outoforder.cc/projects/apache/mod_svn_view/
- [11] SVN::RaWeb::Light, <http://web-cpan.berlios.de/modules/SVN-RaWeb-Light/>
- [12] SVN Browser, <http://www.polarion.org/svnbrowser.php>
- [13] Insurrection, <http://insurrection.tigris.org/>
- [14] CVSGui, <http://www.wincvs.org/>
- [15] Cervisia, <http://www.kde.org/apps/cervisia/>
- [16] TortoiseCVS, <http://www.tortoisecvs.org/>
- [17] Rapid SVN, <http://rapidsvn.tigris.org/>
- [18] Tortoise SVN, <http://tortoisesvn.tigris.org/>
- [19] JSVN, <http://jsvn.alternatecomputing.com/>

A Other SCM Systems

A.1 SCCS

The *Source Code Control System* was originally developed at the Bell Telephone Laboratories. It is a standard part of Unix System V and comes with every flavour of Linux.

A.2 RVS

Revision Control System was developed after SCCS in the early 1980s. It is generally easier to use than SCCS and more powerful. It may also be faster at retrieving a file from the repository though mostly the time difference will not be noticeable.

A.3 Aegis

Aegis (<http://aegis.sourceforge.net/>) supervises the development of changes to a project and the integration of those changes back into the master source of the project. It does not allow changes to be unconditionally added to the master source. It enforces a number of requirements, each designed to ensure that the project does not ‘go backwards’ because of a change. The philosophy of Aegis is to ensure that the source code in the repository (the *Baseline* in Aegis-speak) will build whenever it is checked out. No changes are allowed without passing a build test as well as a functionality test and a code review. Over time the functionality tests will build into a regression test suite for the code.

A.4 Perforce

Vendor

Perforce Software Inc. <http://www.perforce.com/>

Description

The Perforce SCM System features comprehensive software configuration management capabilities built around a scalable client/server architecture. Requiring only TCP/IP, developers can access the Perforce Server through a variety of Perforce clients (GUIs for several platforms, Web, or Command-Line). Perforce can be deployed quickly and easily, and requires minimal administration, even for large or distributed sites.

Available on over 50 operating systems, Perforce includes version control, workspace management, atomic change transactions and a powerful branching model to develop and maintain multiple codelines.

All Perforce software is free and fully functional, with the exception of the Perforce Server, which allows only two users and two client workspaces when used without a license. A Perforce license enables the Perforce Server to support more users and an unlimited number of workspaces. It also entitles you to Perforce Technical Support. Prices are available from the Perforce web site along with details of educational discounts and free licensing for open source development.

A.5 Visual SourceSafe

Vendor

Microsoft <http://msdn.microsoft.com/vstudio/previous/ssafe/>

Description

Through project-oriented version control and rich integration with Visual Studio[®], Visual SourceSafe provides individual developers and small development teams with tools to make safe alterations to existing code and track changes across users, projects, and time.

Visual SourceSafe is a version control system product that delivers restore point and parallel collaboration capabilities, thus allowing application development organizations to work on several versions of software simultaneously. The version control system introduces a check-in and check-out model in which an individual developer checks out a file, makes changes, and then checks the file back in. Other developers typically are not able to make changes to a file while it is checked out. Source code control systems also allow developers to roll back or undo any changes that later create problems.

B CVS Set Up

This section provides a quick run through of the steps I took to perform a basic CVS set up. The set up allows remote connection to the repository via `ssh` (for users with accounts on the server) and web based repository browsing with ViewCVS.

The commands given for remote access assume that the user can `ssh` from the client machine to the server. CVS uses `rsh` by default but this can be changed to `ssh` by setting the environment variable `CVS_RSH` to `ssh`.

Create the Repository

Login to the server (in this case `herodium.cse.rl.ac.uk`) as root and issue the command

```
cvs -d /usr/local/cvsroot init
```

Make sure the the permissions are set so that the people who need to the use the repository have read and write access to the repository directory (`/usr/local/cvsroot` in this case).

Add Files to the Repository

Add the directories and files to put under CVS to a temporary directory (`/tmp/cvs_test`) ensuring that the structure is what you want as it is difficult to move files and directories in the CVS repository.

Import the files as an ordinary user on the server machine

```
cd /tmp/cvs_test
```

```
cvs -d /usr/local/cvsroot -m "Initial import" singch vendor_tag start
```

If the files don't exist on the server machine the import can be done remotely with

```
cd /tmp/cvs_test
```

```
cvs -d :ext:herodium.cse.rl.ac.uk:/usr/local/cvsroot -m "Initial import" \  
singch vendor_tag start
```

In both cases the files and directories are stored in the repository under the name `singch`.

Checkout the Files

By default the files will be checked out to a directory with the same name as the one where the files are stored in the repository so, on the server machine as an ordinary user

```
cvs -d /usr/local/cvsroot checkout singch
```

will put the files in `$cwd/singch`.

On a client machine the files can be checked out with

```
cvs -d :ext:herodium.cse.rl.ac.uk:/usr/local/cvsroot checkout -d new_dir singch
```

Here `new_dir` is the name of the directory to check the files out to.

Repository Browsing with ViewCVS

ViewCVS [5] is a browser interface for CVS and Subversion version control repositories. It generates templatised HTML to present navigable directory, revision, and change log listings. It can display specific versions of files as well as diffs between those versions.

I used the development version of ViewCVS from their CVS repository. Once the code was downloaded I used the following steps to install the software:

1. run `./viewcvs-install` and decide where ViewCVS should be installed,
2. edit the installed `viewcvs.conf` file following instructions in the file,
3. copy `www/cgi/viewcvs.cgi` file to the Apache cgi script directory.

Point your web browser at `http://herodium.cse.rl.ac.uk/cgi-bin/viewcvs.cgi` to view the repositories.

C Subversion Set Up

This section provides a quick run through of the steps I took to perform a basic Subversion set up. The set up allows remote connection to the repository via `ssh` (for users with accounts on the server) or the `http` protocol (for authorised users) and web based repository browsing either with Subversion's own WebDAV functionality, WebSVN or ViewCVS (see above for details).

Create the Repository

On the server (in this case `herodium.cse.rl.ac.uk`) create a directory where the repositories will be kept, e.g. `/usr/local/svn` and make sure that users have read and write access to that directory.

As an ordinary user create a repository

```
svnadmin create /usr/local/svn/singch
```

Set up http Access to the Repositories

To enable `http` access with basic authorisation I followed the following procedure.

1. Edit `/etc/httpd/conf/httpd.conf` to add a `<Location>` for `svn` and map it to `/usr/local/svn`.
2. Edit `/etc/httpd/conf/httpd.conf` to add basic authorisation with `AuthType Basic`
3. Create authentication file `/etc/svn-auth-file` with

```
htpasswd -cm /etc/svn-auth-file djw
```

to add user `djw` and set their password. (The `-c` flag is not required when subsequent users are added since it is the flag to create the file.)

4. Edit `/etc/httpd/conf/httpd.conf` to allow read only activities for anonymous users but require authentication for commits etc.

The Subversion section of `httpd.conf` looks like

```

# Subversion repositories
<Location /svn>
DAV svn

# any "/svn/foo" URL will map to a repository /usr/local/svn/foo
SVNParentPath /usr/local/svn

SVNIndexXSLT "/svnindex.xsl"

AuthType Basic
AuthName "Subversion repository"
AuthUserFile /etc/svn-auth-file

# For any operations other than these, require an authenticated user.
<LimitExcept GET PROPFIND OPTIONS REPORT>
Require valid-user
</LimitExcept>

</Location>

```

Add Files to the Repository

Create a temporary directory, e.g. `/tmp/svn_test` and under that create the directories `branches`, `tags`, and `trunk`.

Add the directories and files to put under Subversion to `svn_test/trunk`.

Import the files as an ordinary user on the server machine

```
svn import /tmp/svn_test file:///usr/local/svn/singch/ -m "Initial import"
```

If the files don't exist on the server machine the import can be done remotely with

```
svn import /tmp/svn_test http://herodium.cse.rl.ac.uk/svn/singch \
-m "Initial import"
```

Checkout the Files

On the server machine as an ordinary user

```
svn checkout file:///usr/local/svn/singch/trunk project
```

will put the files in `$cwd/project`.

On a client machine the files can be checked out with

```
svn checkout http://herodium.cse.rl.ac.uk/svn/singch/trunk project
```

Repository Browsing with Subversion WebDAV

If http access is set up as described above then Subversion's own WebDAV technology will provide access on `http://herodium.cse.rl.ac.uk/svn/singch`.

A stylesheet to format the output can be used with the `SVNIndexXSLT "/svnindex.xsl"` directive to the http server.

Repository Browsing with WebSVN

WebSVN [8] offers a view onto subversion repositories that has been designed to reflect the Subversion methodology. You can view the log of any file or directory and see a list of all the files changed, added or deleted in any given revision. You can also view the differences between two versions of a file so as to see exactly what was changed in a particular revision.

I used version 1.61 of WebSVN the latest at the time of writing. Once the code was downloaded I used the following steps to install the software:

1. unpack the archive to `/usr/local/WebSVN`,
2. Copy `include/distconfig.inc` to `include/config.inc` and edit following instructions in the file.
3. Edit `httpd.conf` to add the line `Alias /WebSVN /usr/local/WebSVN`.

Note that WebSVN has no built in authentication mechanism but the same Apache authentication can be used as when setting up access to the repository via http. The section of `httpd.conf` for WebSVN would then look like

```
Alias /WebSVN /usr/local/WebSVN
<Location /usr/local/WebSVN>

    AuthType Basic
    AuthName "Subversion repository"
    AuthUserFile /etc/svn-auth-file

    Require valid-user
</Location>
```

Point your web browser at `http://herodium.cse.rl.ac.uk/WebSVN/index.php` to view the repositories.

D Converting CVS Repositories to Subversion

The tool `cvs2svn` (<http://cvs2svn.tigris.org>) can be used to convert a CVS repository into a Subversion one. It has many options to control exactly what is transferred but I used only the default options which converts the entire CVS repository, including all tags and branches.

Once it was installed (by running a script as root) converting a CVS repository is simply a matter of issuing the command

```
cvs2svn -s /tmp/singch-svn /usr/local/cvsroot/singch
```

The new repository `/tmp/singch-svn` can then be copied to its correct position (`/usr/local/svn` in the examples above) and will then be available for browsing and checkouts etc.