**Software Engineering (IT350) Assignment - 3**

**Case Study on Version Control Tools**

**Submitted by:** NL Praneetha(15IT127)

**Submitted to:** Raksha Nadgir

**Version Control Tool :**

Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to fix the mistake while minimizing disruption to all the team members.

**Goals of a Version Control System.**

* Allow developers to work simultaneously.
* Do not overwrite each other’s changes.
* Maintain history of every version of everything.

Some of the most popular and most preferred open-source version control systems are :

1. CVS 2. SVN 3. GIT

**1.CVS :**

The Concurrent Versions System (CVS), also known as the Concurrent Versioning System, is a [free software](https://en.wikipedia.org/wiki/Free_software) [client-server](https://en.wikipedia.org/wiki/Client-server) [revision control](https://en.wikipedia.org/wiki/Revision_control) system in the field of [software development](https://en.wikipedia.org/wiki/Software_development). [Dick Grune](https://en.wikipedia.org/wiki/Dick_Grune)  developed CVS as a series of [shell scripts](https://en.wikipedia.org/wiki/Shell_scripts) in July 1986.

Features :

CVS uses a [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server) architecture : A server stores the current version of a [project](https://en.wikipedia.org/wiki/Project) and its history, and clients connect to the server in order to "check out" a complete copy of the project, work on this copy and then later "check in" their changes. Typically, the client and server connect over a [LAN](https://en.wikipedia.org/wiki/Local_area_network) or over the [Internet](https://en.wikipedia.org/wiki/Internet), but client and server may both run on the same machine if CVS has the task of keeping track of the version history of a project with only local developers. The server software normally runs on [Unix](https://en.wikipedia.org/wiki/Unix) (although at least the [CVSNT](https://en.wikipedia.org/wiki/CVSNT) server also supports various versions of [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows)), while CVS clients may run on any major [operating-system](https://en.wikipedia.org/wiki/Operating_system) platform.

Several developers may work on the same project concurrently, each one editing files within their own "working copy" of the project, and sending or checking in their modifications to the server. To avoid conflicts, the server only accepts changes made to the most recent version of a file. Developers are therefore expected to keep their working copy up-to-date by incorporating other people's changes on a regular basis. This task is mostly handled automatically by the CVS client, requiring manual intervention only when an [edit conflict](https://en.wikipedia.org/wiki/Edit_conflict) arises between a checked-in modification and the yet-unchecked local version of a file.

If the check in operation succeeds, then the version numbers of all files involved automatically increment, and the CVS-server writes a user-supplied description line, the date and the author's name to its [log](https://en.wikipedia.org/wiki/Data_logging) files.

One good thing about CVS is that it is not too difficult to learn. It comes with a simple system that ensures revisions and files are kept updated. Given the other options, CVS may be regarded as an older form of technology, as it has been around for some time, it is still incredibly useful for system admins who want to backup and share files.

**2.SVN :**

SVN, or Subversion as it is sometimes called, is generally the version control system that has the widest adoption. Most forms of open-source projects will use Subversion because many other large products such as Ruby, Python Apache, and more use it too. Google Code even uses SVN as a way of exclusively distributing code.  
Because it is so popular, many different clients for Subversion are available. If you use Windows, then Tortoise SVN may be a great browser for editing, viewing and modifying Subversion code bases. If you’re using a MAC, however, then Versions could be your ideal client.

Features :

* Directory versioning : CVS only tracks the history of individual files, but Subversion implements a “virtual” versioned filesystem that tracks changes to whole directory trees over time. Files and directories are versioned.
* True version history : With Subversion, you can add, delete, copy, and rename both files and directories. And every newly added file begins with a fresh, clean history all its own.
* Atomic commits : A collection of modifications either goes into the repository completely, or not at all. This allows developers to construct and commit changes as logical chunks, and prevents problems that can occur when only a portion of a set of changes is successfully sent to the repository.
* Hackability : Subversion has no historical baggage; it is implemented as a collection of shared C libraries with well-defined APIs. This makes Subversion extremely maintainable and usable by other applications and languages.
* Consistent data handling : Subversion expresses file differences using a binary differencing algorithm, which works identically on both text (human-readable) and binary (human-unreadable) files. Both types of files are stored equally compressed in the repository, and differences are transmitted in both directions across the network.

**3.GIT :**

Git is considered to be a newer, and faster emerging star when it comes to version control systems. First developed by the creator of Linux kernel, Linus Torvalds, Git has begun to take the community for web development and system administration by storm, offering a largely different form of control. Here, there is no singular centralized code base that the code can be pulled from, and different branches are responsible for hosting different areas of the code. Other version control systems, such as CVS and SVN, use a centralized control, so that only one master copy of software is used.  
As a fast and efficient system, many system administrators and open-source projects use Git to power their repositories. However it is worth noting that Git is not as easy to learn as SVN or CVS is, which means that beginners may need to steer clear if they’re not willing to invest time to learn the tool.

Features :

* Strong support for non-linear development : Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history. In Git, a core assumption is that a change will be merged more often than it is written, as it is passed around to various reviewers. In Git, branches are very lightweight: a branch is only a reference to one commit. With its parental commits, the full branch structure can be constructed.
* Compatibility with existent systems and protocols **:** Repositories can be published via [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP), [File Transfer Protocol](https://en.wikipedia.org/wiki/File_Transfer_Protocol) (FTP), [rsync](https://en.wikipedia.org/wiki/Rsync) (removed in Git 2.8.0), or a Git protocol over either a plain socket, or [Secure Shell](https://en.wikipedia.org/wiki/Secure_Shell) (ssh). Git also has a CVS server emulation, which enables the use of extant CVS clients and IDE plugins to access Git repositories. [Subversion](https://en.wikipedia.org/wiki/Apache_Subversion) and [svk](https://en.wikipedia.org/wiki/Svk) repositories can be used directly with git-svn.
* Efficient handling of large projects :Torvalds has described Git as being very fast and scalable, and performance tests done by Mozilla[]](https://en.wikipedia.org/wiki/Git#cite_note-33) showed it was an [order of magnitude](https://en.wikipedia.org/wiki/Order_of_magnitude) faster than some version control systems, and fetching version history from a locally stored repository can be one hundred times faster than fetching it from the remote server.
* Cryptographic authentication of history : The Git history is stored in such a way that the ID of a particular version (a commit in Git terms) depends upon the complete development history leading up to that commit. Once it is published, it is not possible to change the old versions without it being noticed. The structure is similar to a [Merkle tree](https://en.wikipedia.org/wiki/Merkle_tree), but with added data at the nodes and leaves. ([Mercurial](https://en.wikipedia.org/wiki/Mercurial) and [Monotone](https://en.wikipedia.org/wiki/Monotone_(software)) also have this property.)