Version Control Tools

-Varshitha M K (15IT151)

Version Control System keeps track of all work an all changes in a set of files and allows many developers to collaborate their work.

CVS (Concurrent Versions System)

It is a free software client-server revision control system used in software development. Server stores the current version of a project and its history, and client has to connect to the server in order to check their project copy, work on this copy and later have to check in their changes. Here client and server connects through LAN or internet but they can run on the same machine if CVS has the task of keeping track of project versions of local developers.

Developers can work on the same project simultaneously and can edit their own "working copy" of the project and checking in their modifications to the server. Server only accepts the recent version of the modified file. Hence developers have to maintain a working copy up to date by incorporating other developer's change on a regular basis. This task is usually handled automatically by the client, requiring manual intervention only when edit conflict arises between a checked-in modification and the yet-unchecked local version of a file.

If the check in operation is successful, then the numbers of all files involved are automatically incremented and server writes a user supplied description line, data and authors name to its log files.

Clients can compare versions, request a complete history change, or check out a historical snapchat of the project as of given version or as of given date. It allows "anonymous read access" where clients can check out and compare versions, only the check-in changes requires a personal account and password.

CVS maintains different branches of a project. For example, a released version of the software project makes one branch, used for bug fixes, while a version under current development, with major changes and new feature can form a separate branch.

CVS uses delta compression for efficient storage of different versions of the same file. This works well the large text files with few changes from one version to the next. This is usually the case for source code files. CVS stores each individual

version on server when it has to be stored as binary because it avoids corruption of binary files.

SVN (Subversion)

It is a software versioning and revision control system. Software developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation.

This system maintains versioning for directories, renames, and file metadata. Users can move and/or copy entire directory-tress very quickly, while retaining full revision history. There is versioning of the symbolic links as well. Renamed/copied/moved/removed files retain full version history.

It has native support for binary files, with space-efficient binary-diff storage. Apache HTTP Server is used as network server, WebAV/Delta-V is used for protocol. There is an independent server process called svnserve that uses a custom protocol over TCP/IP. It has client-server, layered design. Client/server protocol sends diffs in both directions. The cost is proportional to change size and not with respect to data size.

Subversion offers two types of repository storage:

Berkeley DB (deprecated) - Subversion has some limitations with Berkeley DB usage when a program that accesses the database crashes or terminates forcibly. No data loss or corruption occurs, but the repository remains offline while Berkeley DB replays the journal and cleans up any outstanding locks. The safest way to use Subversion with a Berkeley DB repository involves a single server-process running as one user (instead of through a shared filesystem)

FSFS - It works faster than the Berkeley DB backend on directories with a large number of files and takes less disk space, due to less logging. It became the default data store for new repositories. FSFS stores its contents directly within the operating system's filesystem, rather than a structured system like Berkeley DB.

Layers

Subversion system comprises several libraries arranged as layers. Each performs a specific task and allows developers to create their own tools at the desired level of complexity and specificity.

- FS - The lowest level; it implements the versioned filesystem which stores the user data.

- Repos - Concerned with the repository built up around the filesystem.
- mod_dav_svn - Provides WebDAV/Delta-V access through Apache 2.
- Ra - Handles "repository access", both local and remote.
- Client,Wc - The highest level. It abstracts repository access and provides common client tasks, such as authenticating users or comparing versions.

File System

One can view the Subversion filesystem as "two-dimensional". Two coordinates are used to unambiguously address filesystem items:

- **Path** (regular path of Unix-like OS filesystem)
- **Revision**

Each revision in a Subversion filesystem has its own *root*, which is used to access contents at that revision. Files are stored as links to the most recent change; thus a Subversion repository is quite compact. The system consumes storage space proportional to the number of changes made, not to the number of revisions.

Git

It is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

Characteristics:

- 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.
- Distribute Development- Like Darcs, BitKeeper, Mercurial, SVK, Bazaar, and Monotone, Git gives each developer a local copy of the full development history and changes are copied from one such repository to another. These changes are imported as added development branches, and can be merged in the same way as a locally developed branch.
- Compatibility with existent systems and protocols - Repositories can be published via Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), rsync (removed in Git 2.8.0, or a Git protocol over either a plain socket, or 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 and 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 showed it was an 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.[34]
- 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.
- Toolkit-based design - Git was designed as a set of programs written in C, and several shell scripts that provide wrappers around those programs.
- Pluggable merge strategies - As part of its toolkit design, Git has a well-defined model of an incomplete merge, and it has multiple algorithms for completing it, culminating in telling the user that it is unable to complete the merge automatically and that manual editing is needed.
- Garbage accumulates until collected - Git will automatically perform garbage collection when enough loose objects have been created in the repository.
- Periodic explicit object packing - Git stores each newly created object as a separate file. Although individually compressed, this takes a great deal of space and is inefficient. This is solved by the use of *packs* that store a large number of objects delta-compressed among themselves in one file (or network byte stream) called a packfile. Packs are compressed using the heuristic that files with the same name are probably similar, but do not depend on it for correctness. A corresponding index file is created for each packfile, telling the offset of each object in the packfile. Newly created objects (with newly added history) are still stored as single objects and periodic repacking is needed to maintain space efficiency.
- Another property of Git is that it snapshots directory trees of files.

Version control tool which will used for Time Table Management using Android Project: Git