ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 3; Issue 3; May-June; Page No. 05-08

Emerging Version Control Systems for Agile Content Management

Prof.T.Venkat Narayana Rao[1], Thakkallapally Sneha [2], Chenchu Swetha[3] , D. Sreenivasa Rao[4]

[1]Professor, Dept. of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Ghatkesar, RR District, T.S. India

tvnrbobby@yahoo.com

[2]PG Scholar, Dept. of Computer Science and Engineering, Sreenidhi Institute of Science and Technology,Ghatkesar, RR District, T.S. India

sneha.ramrao@gmail.com

[3]PG Scholar, Dept. of Computer Science and Engineering, Sreenidhi Institute of Science and Technology,Ghatkesar, RR District, T.S. India

chenchu.swetha@gmail.com

[4]Asst.Professor, Dept. of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Ghatkesar, RR District, T.S. India

Abstract:
In computer programming distributed version control or decentralized version control, is also known as distributed revision control which allows software developers to work on a given project without requiring a common network. A piece of software which allows you to record and preserve the history of changes made to directories and files is known as Version control. In most of the cases, you should specify set of files or a directory that should have their changes tracked by version control. This can be done by checking out or by telling the software which of your files you wish to have under version control or cloning a repository from a host. The set of directories or files that are under version control are called as a repository. This paper focus on how classical version control systems can be revamped into emerging agile version control systems and tools.

Keywords: Directory, Repository, Software Management, Version control.

INTRODUCTION:

Version control, also known as Revision Control is the process of management of changes to documents, large web sites, computer programs and other collection of information. Changes in the documents are normally identified by a letter code or number [2][6].

A Version Control system is a repository of files and also contains the files for the source code of computer programs, with permissions. Wherever there is a change that is made to the source, then that changes are tracked, along with person who made that changes and why they made it and also reference where the problem have been fixed or any enhancements introduced by the change.

Version control systems are essential for any form of collaborative, distributed development. Whether it is the history of a Wikipedia page or a very large software development project, Version control supports tracking changes done by the individuals and also has the support for reversing changes and also supports checking the difference between various versions of the file [4].

Version control systems (VCS) are stand-alone applications. Revision control is also enclosed in various types of software such as spreadsheets, and word processors e.g., Google Docs and Sheets and in different content management systems, e.g., Wikipedia's Page history.

Revision control has the capability to revert a document to a previous revision, which is significant for allowing editors to trace each other's edits, defend against vandalism and spamming and correct mistakes.
Software tools for revision control are vital for the organization of multi-developer projects. Sites such as Google Code, SourceForge and Github provide Version control [6].

These sites typically build a set of services around version control like archiving, mailing lists, release downloads, web hosting, bug trackers and build farms [1]. For those projects that do not have resources to maintain their own servers, this functionality is well-suited.

1. Types of Version control Systems

1.1 Local Version Control Systems:

The simplest version-control method is coping files form one directory to another directory. This method of copying the files into another directory is very widespread because it is very simple, but also extremely error prone.

Corresponding author: Prof.T.Venkat Narayana Rao

This simple way has been used extensively by many large software projects. But this approach is overlooked, in which directory you are in and may unintentionally write to the wrong file or else copy files which are not needed. This approach is not efficient as many similar copies of the program have to be saved. And for this, developers require lot of self-discipline [3].

To resolve this issue, programmers developed local Version Control Systems as shown in the figure 1 that has a simple database that keeps track of all the changes to files under revision control. Thereupon, systems to automate all or few of the revision control process have been developed.

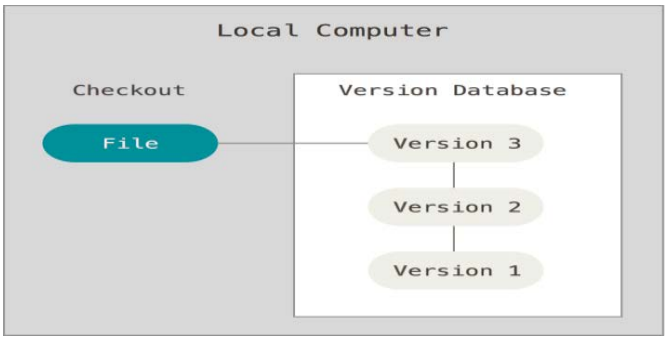

Figure1: Local version control

RCS is one of the popular VCS tools, which is still distributed with many computers today. The Mac OS X operating system also includes the rcs command. Changes are kept as patches and we can get back the files by applying all the patches [5].

1.2 Centralized Version Control Systems:

The next major problem that people encountered is collaboration. That is people need to cooperate with developers on other systems. Therefore to handle this issue, Centralized Version Control Systems (CVCSs) were developed its structure is shown in the figure 2. In these systems, such as Perforce, CVS and Subversion all the changes are kept in a single server and allows the clients to connect to it, and this was the standard followed for the version control systems for a long time [1][5].

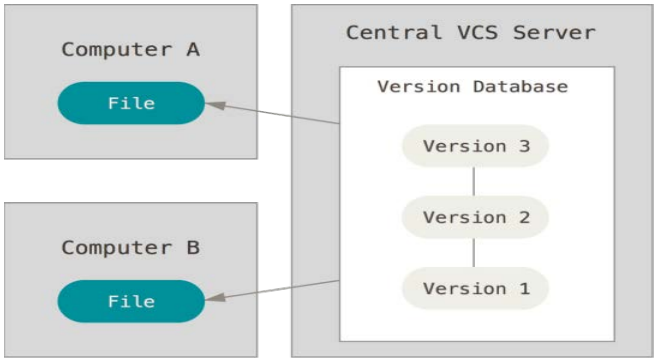

Figure2: Centralized version control

CVS's offers many advantages over local VCSs. For example, everyone in the project knows what everyone else is doing. Administrators have power to control who can do what and it is also easy to administer a CVCS than to deal with local databases.

However, this approach also has some serious drawbacks. The most obvious drawback is single point of failure in the centralized server. If the server goes down for some time, then during that hour nobody can collaborate or save versioned changes. If the hard disk in the central database is corrupted, and backups haven't been kept, everything is lost, the entire history of the project except snapshots that people have on their local machines. A local VCS system where entire history is kept in a single place has the chances of losing everything [5][2].

1.3 Distributed Version Control Systems:

In DVCS, such as Git, Darcs, Bazaar or Mercurial, clients fully mirror the repository and check the latest snapshot of the files.

Thus in case if any server dies, and the systems were collaborating via DVCS, then any of the client repositories can be copied back to the server to restore it. Every clone is a full backup of all the data. The structure of DVCS in shown in figure 3.

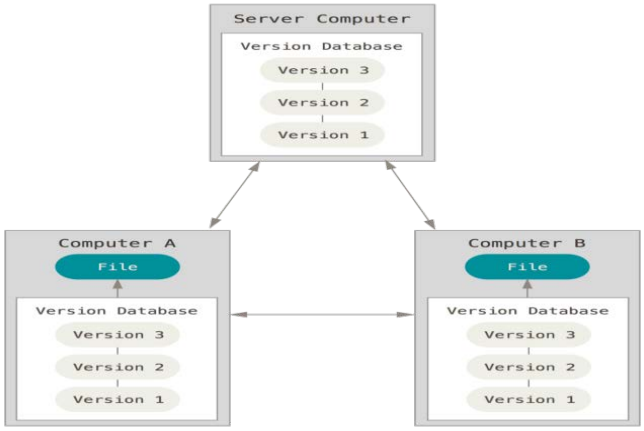

Figure 3: Distributed version control

2. GIT AND MERCURIAL: AN EXAMPLE FOR VERSION CONTROL SYSTEMS:

Git's design was inspired by Monotone and Bit Keeper . Git was originally built as a low-level version control system engine. Others could write front ends on top of Git. Git is very fast, efficient with large projects, and it has incredible branching system[4][3].

The core Git project has entirely became a version control system that is used directly. Some of the goals of Git are: simple design, speed, to provide support for non-linear development (thousands of parallel branches), fully distributed, able to handle large projects like the Linux kernel efficiently (speed and data size). Torvalds was strongly influenced by Bit Keeper, and therefore deliberately avoided conventional approaches, leading to a unique design.

Mercurial was also introduced at the same timeas Git and is also one among the distributed revision control systems. Mercurial was originally proposed to matchup with Git for Linux kernel development. It is different from other revision control systems. Mercurial is opposed to C and is implemented in Python, but there are some instances where C language can be used[2].

Python has distributed nature and is developed using Python. Python language developers are considering a switch to Mercurial because of these features. It will allow developers who do not know core operations, to have easier access to create new trees and reverting changes.

2.1 Major Characteristics of Version Control System:

- Distributed development
- Strong support for non-linear development
- Efficiently handle large projects
- Cryptographic authentication of history
- Git is a collection of many small tools which are written in C, and also contains number of scripts that provide convenient wrappers.
- Presents a generic low-level toolkit for tree history storage and directory content management.
- Compatibility with existing systems/protocols
- Garbage accumulates unless collected
- Periodic explicit object packing

Following are the characteristics of Mercurial:

- Mercurial is very fast and powerful Version Control System.
- Mercurial is easy to learn
- Most of the tasks simply work on the first try without requiring any prior knowledge.

2.2 Data structures:

There are two types of data structures:

- Mutable index also called cache or stage. Mutable index caches information about the next revision to be committed and the working directory.
- Immutable will append-only object database.

Index acts like a connection point between the working tree and the object database in Git. The objects are identified using SHA-1 in both Git and Mercurial. Hash is computed by Git. And this value is used for objects name. The object is placed into a directory where the first two characters of its hash match. And remaining hash is used as objects file name[2].

Each revision of a file is stored as a unique blob in Git. By examining the tree and commit objects we can get the relationships between the blobs. Using zlib compression, newly added objects are stored in their entirety. This process can ingest large amount of disk space; therefore objects can be combined into *packs*. Delta compression is used to save space, and storing blobs as their changes relative to other blobs. Git servers works on TCP port 9418[3].

HTTP-based protocol is used by Mercurial. HTTP protocol see's the number of round-trip requests, new connections and reduces the data movements. Mercurial can also work over ssh where the protocol is very similar to the HTTP-based protocol. 3-way merge is used as default before calling external merge tools.

2.3 Implementations of VCS:

Git is developed on Linux, and also supports most of the operating systems like BSD, OS X, Solaris, and Microsoft Windows. JGit implementation of Git is a pure Java software library which is designed to be embedded in any Java application. In the Gerritcode review tool JGit is used and in EGit, a Git client for the Eclipse IDE. The Dulwich implementation of Git is a pure Python software component for Python 2.

The libGit2 implementation of Git is an ANSI C software library with no dependencies, which can be built on different platforms including Microsoft Windows, Mac OS X, Linux, and BSD. It supports interfacing with different programming languages [4].

Mercurial is implemented in python. Mercurial is distributed in nature. Mercurial creation in Python, therefore most of the Python developers are considering a switch to Mercurial from Git, as it would allow easier access in creating new trees and reverting changes for non-core developers

Mercurial have some features which are similar to SVN, and because of the similarities, learning Mercurial will be easy for those who are already familiar with SVN. Mercurial documentation is also complete and helps to learn the differences faster [1].

Some of the drawbacks in Mercurial include that unlike Git, mercurial does not allow two parents to be merged, rather than being scriptable it uses extension systems.

2.4 Advantages and Disadvantages:

Advantages of Git:

- Those who hate CVS/SVN, Git can be used
- Operation speed is increased
- Cheap branch operations
- Distributed, peer-to-peer model
- Full history tree available offline

Disadvantages of Git:

- Not optimal for single developers
- Compared to Linux Git has limited Windows support

Advantages of Mercurial:

- Better documentation
- Distributed model
- It is easy to learn compared to Git

Disadvantages of Mercurial:

- Based on extension
- Less out of the box power
- merging of two parents is not allowed

3. CONCLUSION:

Over the years many backup schemes and version control systems were used. They are having all facilities for retrieving the past contents of a file. Most of the version control systems have methods to show how a file varies over the time. Many of these systems allows you to get into past, begin a distinct line of reasoning, and then bring the new thoughts back to the present. Still fewer systems offer fine-grained control over that process, which allows you to collect your thoughts moreover you feel fine to present your ideas to the public. Git allows you do all the things, and with less difficulty, once you get aware of its fundamentals. It is not the only system which employ the best interface to its. Version Control system is useful because it allows to examine the changes which resulted in each of those versions and facilitates to extract particular version of the file.

4. REFERENCES:

1. Fung, K. H., Aurum, A., Tang, D. (2012). Social Forking in Open Source Software: An Empirical Study, Proceedings of the CAiSE'12 Forum at the 24th International Conference on Advanced Information Systems Engineering (CAiSE), 855(1), 50-57.
2. K. Herzig and A. Zeller. The impact of tangled code changes. In Proceedings of the Tenth International Workshop on Mining Software Repositories, pages121130. IEEE Press, 2013.
3. G. Gousios and D. Spinellis. Ghtorrent: Github's data from a _rehose. In Mining Software Repositories(MSR), 2012 9th IEEE Working Conference on, pages 12{21. IEEE, 2012.
4. Orsila, H., Geldenhuys, J., Ruokonen, A., Hammouda, I. (2009). Trust issues in open source software development. International Conference on Software Engineering, Proceedings of the Warm Up Workshop for ACM/IEEE ICSE 2010, Cape Town, South Africa, 9-12.
5. Oezbek, C., Thiel, F. (2010). Radicality and the open source development model, Proceedings of the FLOSS Workshop 2010.
6. C. Bird, A. Bachmann, F. Rahman, and A. Bernstein. Linkster: enabling e_cient manual inspection and annotation of mined data. In Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, pages 369{370. ACM, 2010.