IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-12, Dec 2018 E-ISSN: 2347-2693

Migration from Subversion to Git Version Control System

Monika Varshney[1*], Azad Kumar Shrivastava[2], Alok Aggarwal[3], Adarsh Kumar[3]

[1,2]Department of Computer Science, Mewar University, Chittorgarh (Raj), India
[3]School of Computer Science, University of Petroleum & Energy Studies, Dehradun, India

[*]*Corresponding Author:* monikafpc@gmail.com

Available online at: www.ijcseonline.org

Accepted: 26/Dec/2018, Published: 31/Dec/2018

***Abstract* -** In recent years, software development in software industries is getting a transition from centralized version control systems (CVCSs) like subversion, mercurial, perforce, CVS etc. to decentralized version control systems (DVCSs) like Git due to a number of reasons like time, space, branching, merging, offline commits & builds and repository etc. Both centralized VCSs and distributed VCSs have gone through ample investigations in recent past but individually from the software developer's point of view in a large commercial software industry. There has been a little focus on the transition across Git having a share of more than three-fourth of total VCS, and Subversion having a share of 13.5%. In this work transition process from Subversion VCS to Git VCS has been investigated.

***Keywords* -** Version control system, distributed VCS, centralized VCS, transition, branching, merging, time, space

I. INTRODUCTION

An enormous growth has been witnessed during the last five decades by computers. Computers have evolved from very expensive hall sized systems to large mainframes used by fortune 500 companies, to the advent of affordable personal computers, to the most recent mobile and cloud computing on minimal devices. Memory and storage have always been a very vital parts of any computer system and as systems have modernized, file systems have been created and evolved in order to keep up. Version control is often used by software developers that helps a software team to manage changes to source code over time. In a special kind of database, a trace is kept of every modification in the software by a Version Control System (VCS).

VCS has a very long history in computing starting from the source code control system, developed in 1972. Now-a-days VCS are quite popular and becoming essential so with the advent of cloud based systems, such as Github, Bitbucket etc. For almost all software projects, the source code is like the crown jewels - a precious asset whose value must be protected. For most software teams, the source code is a repository of the invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort. Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.

Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized in a folder structure or "file tree". One developer on the team may be working on a new feature while another developer fixes an unrelated bug by changing code, each developer may make their changes in several parts of the file tree. While it is possible to develop software without using any version control, doing so subjects the project to a huge risk that no professional team would be advised to accept. Good version control software supports a developer's preferred workflow without imposing one particular way of working. Ideally it also works on any platform, rather than dictate what operating system or tool chain developers must use. Great version control systems facilitate a smooth and continuous flow of changes to the code rather than the frustrating and clumsy mechanism of file locking - giving the green light to one developer at the expense of blocking the progress of others.

Software teams that do not use any form of version control often run into problems like not knowing which changes that have been made are available to users or the creation of incompatible changes between two unrelated pieces of work that must then be painstakingly untangled and reworked. If you're a developer who has never used version control you may have added versions to your files, perhaps with suffixes like "final" or "latest" and then had to later deal with a new final version. Perhaps you've commented out code blocks because you want to disable certain functionality without

deleting the code, fearing that there may be a use for it later. Version control is a way out of these problems.
Regardless of what they are called, or which system is used, the primary benefits you should expect from version control are as follows.

- A complete long-term change history of every file
- Branching and merging
- Traceability

In this work transition process from Subversion VCS to Git VCS has been from software developer point of view working individually or in a large team over a large and complex software having a legacy of many decades. Rest the paper is organized as follows. Related work done by earlier researchers is reported in section 2. Section 3 gives VCS with Git. Industry based tools for SVN to Git migration are given in section 4. Work is finally concluded in section 5.

II. RELATED WORK

Although almost nothing in the published research work was found concerning transition of Subversion VCS to Git VCS with respect to time, space branching, merging and repository aspects, there is much published research work on the other version control systems [1]-[5], work which uses Git as a file system [6]-[9]. In [9] transition from CVCS to DVCS is investigated based on interviews and survey data and some guidelines are proposed for the individual developers, teams and managers who consider transitioning. This work identifies ability to work offline and incrementally and managing multiple contexts efficiently as the major transition expectations which are satisfied by commits and lightweight branches, available on most DVCSs. In [10] authors have investigated the transition process, challenges and anticipated benefits of four open source software. In [11] an investigation has been done on the way the developers use branches in an open source software and how the transition affects the project branching structure. A system has been described in [7] which exposes the provenance stored in VCS which enables the easy publication of VCS provenance on the web and subsequent integration with other systems that make use of PROV.

III. VERSION CONTROL WITH GIT

There are various version control systems are being used. Example: SVN, Mercurial, VCS, GIT and many more. By far, the most widely used modern version control system in the world today is Git. Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments).

Google:
"Today, Git holds a commanding share of the VCS market based on Google Trends data. However, Apache Subversion and Mercurial are still used in many environments."

Figure 1 shows the data collected from software engineers working on Version Control Systems for the four years and shows the percentage of different used VCSs. Figure 2 gives the web search interest share of top 5 VCS in 2016.

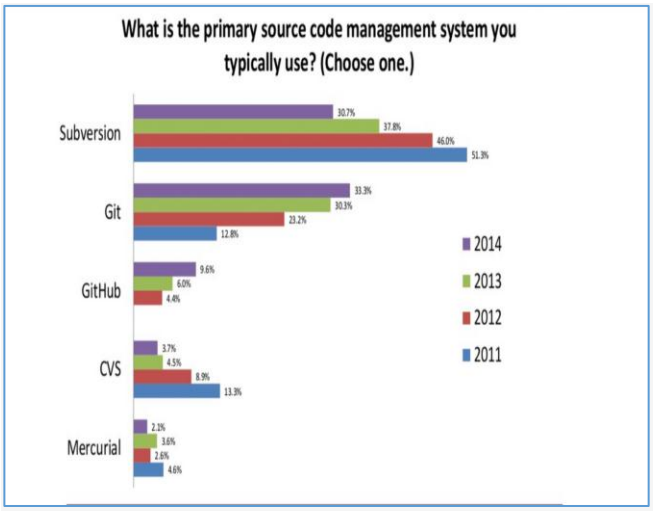

Figure 1: Data Collected from different software engineers working on Version Control Systems for the four years

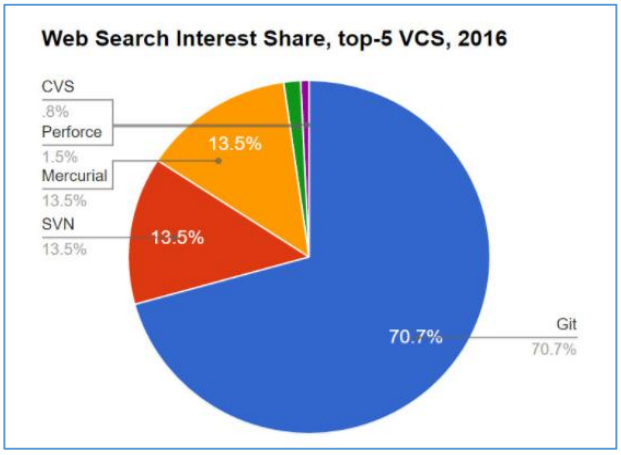

Figure 2: Web search interest share of top 5 VCS in 2016

Having a distributed architecture, Git is an example of a DVCS (Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

In addition to being distributed, Git has been designed with performance, security and flexibility in mind. How Git is better than other VCS in terms of command difrence is shown in Table 1. Table 2 shows the different properties of CVS, SVN and Git. Figure 3 gives a direct comparison of Git with SVN.

Table 1: Command difference among major VCSs

software	clone	pull	push	checkout	update	merge	revert
Bazaar	branch	pull	push	checkout	update	merge	revert
CVS	N/A	N/A	N/A	checkout	update	update -j	remove [then] update
darcs	get / put	pull	push	get	pull	pull / push	revert
Git	clone	fetch	push	clone	pull	merge	checkout
Mercurial	clone	pull	push	clone	pull -u	merge	revert
Subversion	svnadmin hotcopy	[work-around]: svnadmin load	[work-around]: svnadmin dump	checkout	update	merge	revert

Table 2: Different properties of CVS, SVN and Git

CATEGORY	CVS	SVN	GIT
Distributed/Centralized	Centralized	Centralized	Distributed
Initial Release	1985	2001	2015
License	GNU GPL	GNU GPL	GNU GPL
Version	1.11.7	1.5.6	1.6.1
OS	Windows, OS X, UNIX	Windows, OS X, UNIX	Windows (limited), OS X, UNIX
Atomic commits	No	Yes	Yes
Move and renames without losing the history	No	Yes (some exceptions)	Yes (some exceptions)
Partial checkouts	Yes	Yes	No
Tracking of branches and merges	No	Yes	Yes
Revision numbers	Increasing integer	Increasing integer	SHAI
Local repository	No	No	Yes
Central repository	Necessary	Necessary	Available
Workflow	Static	Static	Flexible
User interface	Sometimes confusing	Good	Rather complex
Consistency check	No	No	Yes (SHAI)
Tagging	Expensive	Cheap	Cheap
Branches	Expensive	Cheap	Cheap
Performance	Very slow	Slow	Very good
Recognizes binary files	Yes	Yes	Yes
Handling of binary files	No diff	Diff	Diff
Compression of repository	No	No	Yes
Two phase locking	Available	Available	No
Path based authentication	Yes	Yes	No

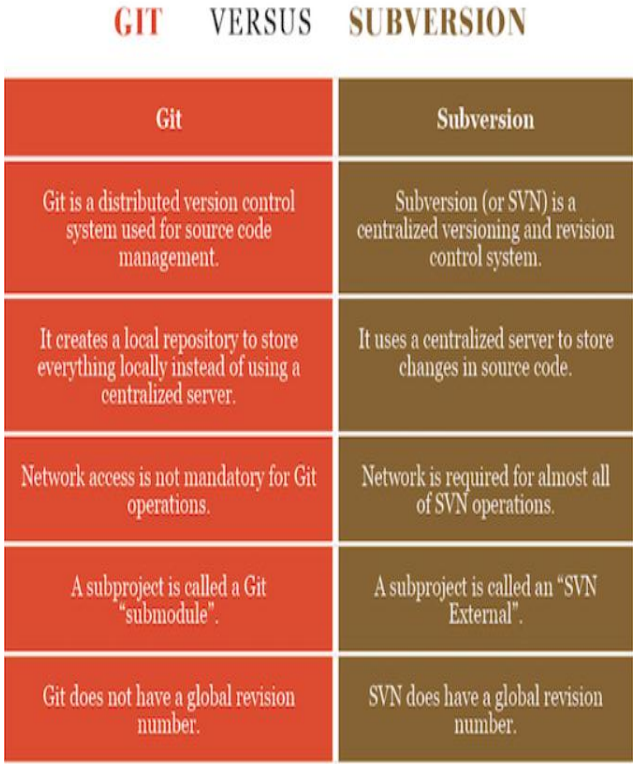

Figure 3: Direct comparison of Git with SVN

IV. INDUSTRY BASED TOOLS: SVN TO GIT MIGRATION

There are many tools available in markets to migrate SVN to Git but leading tools are SVN2Git and SubGit.

V. CONCLUSION

Version control systems protect source code from both catastrophe and the casual degradation of human error and

unintended consequences. Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized in a folder structure or "file tree." One developer in the team may be working on a new feature while another developer fixes an unrelated bug by changing code, each developer may make their changes in several parts of the file tree. In this work transition process from Subversion VCS to Git VCS has been from software developer point of view working individually or in a large team over a large and complex software having a legacy of many decades.

REFERENCES

[1]. N. B. Ruparelia. "The history of version control," *ACM SIGSOFT Software Engineering Notes* vol. 35, no. 1, pp. 5-9, 2010.

[2]. B. De Alwis and J. Sillito, "Why are software projects moving from centralized to decentralized version control systems?" *ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE'09)*, pp. 36-39, 2009.

[3]. D. Spinellis. "Git," *IEEE Software*, vol. 29, no. 3, pp. 100-101, 2012.

[4]. https://www2.physics.ox.ac.uk/it-services/moving-projects-from-svn-to-git

[5]. https://stosb.com/static/talks/case_study_git_efl_linuxcon_eu_13.pdf

[6]. Loeliger, J., Matthew McCullough, "Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development," O'Reilly Media, Inc. Second Edition, 2009.

[7]. Tom De Nies, Sara Magliacane, Ruben Verborgh, Sam Coppens, Paul Groth, Erik Mannens, and Rik Van de Walle, "Git2PROV: Exposing Version Control System Content as W3C PROV," *Proc. 12th Int. Semantic Web Conf.*, pp. 1-4, Oct. 2013.

[8]. C. Brindescu, M. Codoban, S. Shmarkatiuk and D. Dig, "How Do Centralized and Distributed Version Control Systems Impact Software Changes?," *Proc. 36th Int. Conf. on Software Engineering*, Hyderabad, India, pp. 322-333, 2014.

[9]. Kıvanç Muşlu, Christian Bird, Nachiappan Nagappan, Christian Bird, "Transition from Centralized to Decentralized Version Control Systems: A Case Study on Reasons, Barriers, and Outcomes," Proc. Int. Conf. on Software Engineering ICSE-2014, pp. 334-344, May 31- June 7, Hyderabad, India, 2014.

[10]. Christian Bird, Peter C. Rigby, Earl T. Barr, David J. Hamilton, Daniel M. German, Prem Devanbu, "The Promises and Perils of Mining Git," Proc. 6th IEEE International Working Conference on Mining Software Repositories, Vancouver, BC, Canada, May 16-17, pp. 1-10, 2009

[11]. E. T. Barr, C. Bird, P. C. Rigby, A. Hindle, D. M. German and D. Premkumar, "Cohesive and Isolated Development with Branches," *Proc. 15th International Conference on Fundamental Approaches to Software Engineering*, Tallinn, Estonia, pp. 316-331, March 24 - April 01, 2012.

Authors Profile

Monika Varshney is an Assistant Professor with Dr. Bhimrao Ambedkar University, Agra, India and enrolled in Ph.D. (C.S.E.) from , Mewar University, Gangar, Chittorgarh (Raj) India. She received her M.C.A. from IGNOU, New Delhi, India in the year 2008. Her research interest includes Data mining, Data Base Management System, Algorithm development and Decision Support System etc.

Azad Shrivastava is Professor at Department of Computer Science, Mewar University, Gangar, Chittorgarh (Raj) India. He did his Ph.D. from 'Atal Behari Vajpayee-Indian Institute of Information Technology and Management', Gwalior, Madhya Pradesh, India in the year 2009. He has an academic, research, and industry experience of about 14 years. He has been associated with CMC Ltd., TCS, AETPL. His areas of interest include Deep Learning, Machine learning, AI and NN & Big data on CPU & GPU Cluster for DWH & IOT etc.

Alok Aggarwal received his bachelors' and masters' degrees in Computer Science & Engineering in 1995 and 2001 respectively and his PhD degree in Engineering from IIT Roorkee, Roorkee, India in 2010. He has academic experience of 18 years, industry experience of 4 years and research experience of 5 years. He has contributed more than 150 research contributions in different journals and conference proceedings. Currently he is working with University of Petroleum & Energy Studies, Dehradun, India as Professor in CSE department.

Dr. Adarsh Kumar received his Master degree (M. Tech) in Software Engineering from Thapar University, Patiala, Punjab, India, in 2005 and earned his PhD degree from Jaypee Institute of Information Technology University, Noida, India in 2016 followed by Post-Doc from Software Research Institute, Athlone Institute of Technology, Ireland during 2016-2018. Currently, he is working with University of Petroleum & Energy Studies, Dehradun, India as Associate Professor in School of Computer Science.

