Configuration Management: A comparative Analysis of CVS and SVN

Arnold Mashud Abukari
System/Telecom Engineer, Sunberry Corporations FZE
amashud@sunberrycorp.com

ABSTRACT: ***The growth of information systems is at a very fast pace in both developed and developing countries. A major driving force in the field of Information Technology is software. This software largely controls the hardware components of systems geared towards increasing productivity within an establishment. There is the need for companies and software developers to keep track of the configuration changes on their source codes. This has given rise to the need for proper configuration management in the field software engineering by developers and establishments with so much interest in their source codes. The area of software engineering responsible for controlling software evolution from a source code is configuration management (CM). Configuration management (CM) is a systems engineering process for establishing and maintaining consistency of a product's performance, functional and physical attributes with its requirements, design and operational information throughout its life. The intent of this paper is to analyze configuration management through a comparative analysis of the Concurrent Version System (CVS) and Subversions (SVN). There has been a report of substantial advantages of SVN over CVS in terms of configuration management (CM). This paper also seeks to clarify such reports by making a comparative analysis between the two software versioning and revision control systems.***

KEYWORDS: Configuration management, CVS, SVN

I. INTRODUCTION

It is very essential in the field of Information Technology (IT) industry, telecommunication industry, manufacturing industry and all industries dealing with processes and reviews of products to inculcate configuration management in their practices. This paper intends to highlight configuration management in software engineering taking telescopic eye on the Concurrent Version System (CVS) and Subversions (SVN).

A. CONFIGURATION MANAGEMENT

Configuration management is a collection of processes and tools that promote consistency, monitor changes, and provide updates and consistent documentation and visibility. By building and maintaining configuration management best-practices, one can expect several benefits such as improved product availability and lower costs. These products could be software, networking, database and hardware related. Lack of best practices in configuration management (CM) on a product could result to one or more of the following:

- Inability to monitor user impacts on product changes
- Time to resolve problems from such products is usually high
- Higher product cost because of unused product components
- Lower availability and increased reactive support issues

The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle. Software Configuration Management involves identifying configuration items for the software project, controlling these configuration items and changes to them, and recording and reporting status and change activity for these configuration items [7].

In product development like the software, changes are made to correct errors and/or to provide enhancements to the core products. These changes could come as a result of different contributions from different people working on the same product. There is therefore the need to institute control and merge the inputs made by the various contributors whiles having the option to revert back to a previous state of the product when changes appear to have an adverse effect on the product. This quest of keeping the inevitable changes under control requires an efficient Configuration Management system (CM).

Every organization requires standards and well structured policies to regulate how changes are done

using a well thought-through Configuration Management Plan (CMP).

Academic research and industry analysis show that software CM (SCM) is clearly evolving in ways that will support the needs of product lines. Westfechtel and Conradi, observing the overlap between software architecture and SCM, describe five approaches for the integration of the two disciplines [2].

In the field of Industry, Schwaber at Forrester Research observes that today's SCM market encompasses a range of four solution segments of cumulatively increasing functionality: (1) version control, (2) software configuration management, (3) process-centric software configuration management, and (4) application life-cycle management [3]. According to Schwaber and colleagues, all the major vendors now offer process-centric SCM solutions, and there is growing interest in expanding SCM into application life-cycle management.

The IEEE/ANSI came out with a detailed standard to outline a comprehensive configuration management plan (CMP) in their IEEE 1987a [4]. These plans contain change control policies, describe organizational roles, define product life cycles, and, in general, make a fine starting point for an organization wishing to craft its own CM plan.

B. CONFIGURATION MANAGEMENT PROCESS FLOW

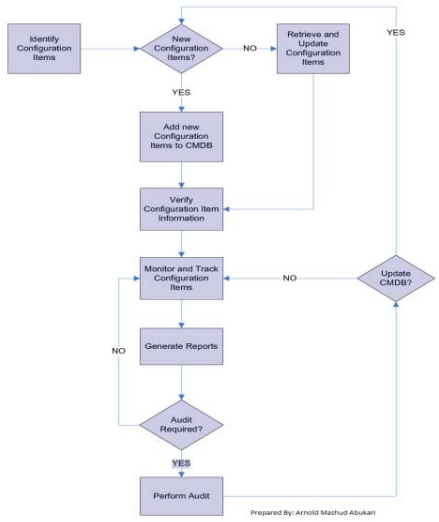

Figure 1: Configuration management process flow

II. CONCURRENT VERSION SYSTEM

The Concurrent Versioning System (CVS) is a type of version control system. A version control system keeps track of all activities and changes made in a set of files or project in the field of software development or engineering. This allows developers across networks or locations to work in a collaborative way towards completing the project.

CVS allows multiple people to be working on the same set of assets at the same time. CVS works on the principle of Copy-Modify-Merge principle. This means you take a copy of the assets, modify your local copy, and merge your changes back into the master repository. The Concurrent Versioning System was developed by Dick Grune as shell scripts in July 1986. [11]

A. CVS WORKING PROCESSES

The Concurrent version systems follow a working process to enable the developers contribute in a collaborative manner. Below are the processes involved:

- Creating a Repository (one time only)
- Importing Assets
- Checking out a working copy
- Viewing changes
- Committing changes
- Working with previous versions

B. COMMANDS FOR CREATING CVS REPOSITORY

A working repository is the backbone of every project that needs review from developers. In CVS, the repository are created using the below set of commands:

```
% export CVSROOT=~/CVSRoot

% cvs init

% ls –l ${CVSROOT}

        drwx------   CVSROOT/
```

After a successful creation of the repository, there is a need to import assets. below is an example of a command for importing an asset:

$ cvs import -m "Imported sources" arnold/*dir* filename start

It is interesting to note that importing the assets doesn't touch the files in the current directory and as such changes in the working directory are not tracked.

C. CVS DEVELOPMENT STATUS

The latest release for CVS came out on 8th May, 2008 and since then there has been maintenance to fix bugs in the CVS repository [10]. However, there has not been reports on bugs since the last update on CVS.

D. CVS SUCESSORS

CVS is reported to have been developed from a versioning control system called Revision Control System (RCS) which manages individual files but not projects. Though CVS came to solve this by managing projects, developers still require additional features that CVS does not provide, needs to alter the operational models in CVS as well as improve developer productivity. These interests have given rise to the phrase YACC (Yet Another CVS Clone). CVS replacement attempts projects include CVSNT, Subversions, EVS and OpenCVS [1].

E. CVS CRITICISMS

There has been debates from critics of the Concurrent Version System since the emergence of SVN and other version control systems. However, admirers of CVS usually take exceptions to some criticisms meted out to CVS. SVN is seen to be a replacement for CVS. Below are some criticisms of CVS:

- Branch operations are expensive.
- Commits are not atomic.
- CVS treats files as text by default.
- No support for distributed revision control or unpublished changes.
- Support for Unicode and non-ASCII filenames is limited.
- No version of symbolic links.
- Revisions created by a commit are per file, rather than spanning the collection of files that make up the project or spanning the entire repository.
- CVS does not version the moving or renaming of files and directories.

III. SUBVERSIONS (SVN)

Subversions as abbreviated as SVN is a software versioning and revision control system (RCS) distributed under the Open Source License. SVN are also used to maintain current and historical events made by the developers on source codes, web pages, databases etc. Subversions are arguably seen as a replacement for the Concurrent Versioning System (CVS). Some projects that are executed using SVN are Apache Software Foundation, Free Pascal, FreeBSD, Mono, GCC and SourceForge. A report by Forrester Research in 2007 indicates that Subversions (SVN) is the sole leader in the standalone Software Configuration Management (SCM) category as well as a strong performer in the Software Configuration and Change Management (SCCM) category [8].

In 2000, CollabNet Inc. created Subversions (SVN) and it's now a top level Apache project being built by a community of developers. Subversion lead by example by hosting its own source codes in 2001 to enable contributors to add value to it as a project as reported by Collins-Sussman et al [2]. Subversions were accepted into an Apache Incubator in November, 2009 and later became a top level Apache project in February, 2010 [5].

A. SUBVERSIONS REPOSITORY TYPES

A repository commonly refers to a location for storing often used for preservation or safety of codes. SVN has two types of repository storage namely Berkeley DB and FSFS (File System File System).

a. Berkeley DB

The Berkeley DB package is originally used in developing SVN. However, some limitations were realized when a program crashes, the Repository goes offline even when no data is lost and no corruption of data occurs. This means that the safest way to use the Subversions with Berkeley DB is a single server process used as a single user as indicated by Ben Collins-Sussman et al [3].

b. FSFS

FSFS was developed as a new storage subsystem in 2004. Ben Collins-Sussman reports that, FSFS works faster than Berkeley DB on file directories and take less disk space compared to the Berkeley DB due to less logging [3]. FSFS became the default data repository beginning from subversions 1.2.

c. Repository Access

Gaining access to subversion repository can occur in three ways as indicated below:

- Through a local filesystem or network file system
- Over http/https using the mod_dav_svn for apache 2
- over TCP/IP using the custom "svn" protocol with a default port of 3690

All above three ways can be used to access both the Berkeley DB and the FSFS repositories in subversions.

IV. SIMILARITIES BETWEEN CONCURRENT VERSION SYSTEM(CVS) AND SUBVERSIONS (SVN)

Despite the debate surrounding CVS and SVN there are still some similarities that exist between those two (2) systems. The Analysis of some identified similarities are outlined below:

- CVS and SVN both have the ability to work on only one directory of the repository
- CVS and SVN can both track uncommitted changes by using "cvs diff" and "svn diff" respectively.
- Both systems have no way to assign a per-file commit message to the changeset as well as per-changeset message. Only CVS commit messages are per-change. SVN has no such feature.
- Both systems support web interfaces
- CVS and SVN both have available Graphical User Interface (GUI)
- CVS and SVN are both open source under GNU GPL and Apache/BSD-Style licence.
- They both have the ability to track the history of codes line-by-line using "cvs annotate" and "svn blame" commands.
- They both have networking support using ssh

A. COMPARATIVE DIFFERENCES OF CVS AND SVN

The comparative differences between the two systems will highlight some key important features as well as give indication of a better system in the following categories:

- Repository format
- Speed
- Tags and branches
- Availability
- Meta data
- File types
- Roll back ability
- Transactions
- Internal Architecture and code
- Networking support
- Per change commits messages feature

V. REPOSITORY FORMAT

The Repository format is used to automatically create configuration files for new repositories whose information is published in unavailable format repositories, which can be installed using packages. CVS has a better repository format compared to SVN within the context below:

CVS	This is based on RCS (revision control system) files of versions control. Each file connected to CVS is an ordinary file containing some additional information. It is quite natural that the tree of these files repeats the file tree in the local directory. It is worth noting that, with CVS you can easily correct RCS files if necessary.
SVN	SVN is a relational database (BerkleyDB) and FSFS. On one hand, this settles many problems (for example, concurrent access through the file share) and enables new functionalities (for example, transactions at operations performance). However, on the other hand, data storage now is not transparent, or at least is not available for user interference. That is why the utilities for "curing" and "recovering" of the repository (database) are provided.

A. SPEED

Speed in configuration management is how fast an activity could be executed to bring out the expected output. Speed is very essential when developing software in a community of developers. In the context of system design, SVN is faster than CVS. SVN transmits less information through the network and supports more operations for offline mode.

B. TAGS AND BRANCHES

A tag or branch is nothing other than a copy of a specific revision of your project folder located in a known location in the repository. However, a tag is completely different from a branch. A tag is also called a label in other version control systems, a tag uniquely identifies the state of a part or all of the repository at a particular point in time whiles a branch is a copy of an existing folder at a given point in time. A branch is technically identical to a tag. The two only really differ in terms of intent, i.e. what, by convention, they are intended for. While a tag is generally intended as a static snapshot, a branch is intended to be continued to be modified subsequent to the branch operation. This allows two versions of a folder starting from a common point to diverge over time independently of one another.

Tags and branches are implemented properly in the Concurrent Version System (CVS) compared to Subversions (SVN). In SVN, both tag creation and branch creation are substituted for copying within the repository. From the SVN developer's viewpoint, this is very elegant decision, which simplifies one's life. However, the ability to tag a code is missing in SVN and has been compensated by using the Universal Numbering of files. This means that the whole repository gets the version number, but not each separate file. It is not very convenient to store a four-digit number instead of a symbolic tag.

C. AVAILABILITY

CVS is more available compared to SVN in the context below:

CVS	Presently CVS is supported everywhere where you might need it.
SVN	SVN not yet so widely used, as the result there are places where it supports still not implemented.

D. METADATA

Metadata is simply "data about data" or can also be defined as data providing information about one or more aspects of the data. SVN has a better metadata compared to CVS within the context outlined:

CVS	Store only files and nothing else.
SVN	SVN stores files as well as attach files.

E. FILE TYPES

The file type is the standard way a file is encoded in a computer. It indicates how bits are used to encode files in a digital storage medium. SVN again has a better file type against CVS.

CVS	CVS was initially intended for text data storage. That is why storage of other files (binary, unicode) is not trivial and requires special information, as well as adjustments on either server or client sides.
SVN	SVN manipulates all the file types and does not require further instructions.

F. ROLLBACK ABILITY

Rollback is an operation which returns an activity to some previous state after committing that activity usually in database and software related issues. CVS has the ability to rollback a committed activity hence making it better compared to SVN in this context since SVN has no such feature.

CVS	CVS can roll back any commit in the repository, even if this may require some time (each file should be processed independently)
SVN	SVN does not allow rollback of the commit. The authors suggest copy good repository state to the end of a trunk to overwrite bad commit. However bad commit itself will remain in the repository.

G. TRANSACTIONS

A transaction comprises a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. It is interesting to note that Subversions operate on the principle "all or nothing" when dealing with data transfer or transaction. This feature gives it an edge over CVS.

H. INTERNAL ARCHITECTURE AND CODE

CVS was written initially as scripts around RCS executables but later was constituted into a single executable codes. It has poor internal structure of codes and historical bug fixes. SVN has a better internal structure and codes. SVN codes are expandable and have room for future improvements.

I. NETWORKING SUPPORT

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. 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. The above analysis makes SVN a little better compared to CVS in dealing with networking support.

J. COMMIT MESSAGES FEATURE

Commit messages tell us the status of our transactions. It is important to know the status of your commit command to enable you keep track our operations. CVS has a pre commit message feature to alert developers the status of their operations whiles SVN has no such feature. This makes CVS a better candidate when a developer requires commit messages to keep track of his operations.

VI. CONCLUSION

I have presented an overview of configuration management and outlined the similarities between Concurrent Version System (CVS) and Subversions (SVN). A comparative analysis of some of their differences was also highlighted and a better candidate in each category indicated.

This paper provides an understanding of configuration management comparing CVS and SVN to broaden the understanding in those two versioning and control systems. Contrary to the view by many that SVN is a replacement of CVS, this paper highlights different categories that see each of the two systems as a better option compared to the other. The choice of a versioning and control system solely lies on the priority of the developer on the features of each system. Thus, per the analysis above, SVN cannot be considered a CVS substitute. It is a different system, similar to CVS. It has unique functions, which can serve as a reason for its use. These functions make it more suitable for some development environments.

VII. REFERENCES

[1] Collins-Sussman, B. "Dispelling Subversion FUD". Retrieved June 30, 2010.[Online] Available:(http://www.red-bean.com/sussman/svn-anti-fud.html)

[2] Collins-Sussman, B., Brian W., Fitzpatrick, C., Pilato M. (2011) "What is Subversion? > Subversion's History". *Version Control with Subversion (for Subversion 1.7)*. Retrieved 15 March 2012.

[3] Collins-Sussman, B., Brian W., Fitzpatrick, C., Pilato M. (2011) "Chapter 5: Strategies for Repository Deployment". *Version Control with Subversion: For Subversion 1.7*. O'Reilly.

[4] *IEEE Guide to Software Configuration Management* (1987). New York, NY: Institute of Electrical and Electronics Engineers.

[5] Rubinstein, D. (2009) "Subversion joins forces with Apache". SD Times. Retrieved 15 March 2012.

[6] Schwaber, C., Barnett, L., Friedlander, D., Hogan L. (2005) *the Expanding Purview of Software Configuration Management*. [Online]. Available:http://www.forrester.com/Research/Document/Excerpt/0,7211,36337,00.html.

[7] Software Engineering Institute (2000) *Capability Maturity Model Integration, Version 1.1 CMMI for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1)* (CMU/SEI-2000-TR-018, ADA388775). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[8] The Forrester Wave: Software Change and Configuration Management, Q2 (2007) Forrester Research.

[9] Westfechtel, B., Conradi R. (2003) "Software Architecture and Software Configuration Management," *Proceedings of the ICSE Workshops SCM 2001 and SCM*, pp 24-39.

[10] www.download.savannah.gnu.org/releases/cvs/source/stable/1.11.23 "Index of /releases/cvs/source/stable/1.11.23". (Retrieved January 15, 2013).

[11] www.ximbiot.com/CVS/Concurrent Versions System v1.12.12.1/Overview (Retrieved 9 December 2011).