Comparative Analysis of SCM tools and Coexistence with Agile Methodology

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By

Satyam Gupta
(801332022)

Under the supervision of:

Dr. (Mrs.) Rinkle Rani
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, *"Comparative Analysis of SCM tools and Coexistence with Agile Methodology"*, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Rinkle Rani and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Satyam

(Satyam Gupta)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Rinkle 15/07/15

(Dr. Rinkle Rani)
Assistant Professor
Computer Science and
Engineering Department

Countersigned by

(Dr. Deepak Garg)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

Training Certificate

STMicroelectronics Private Ltd.
Plot No.1,Knowledge Park-III
Greater Noida – 201 308
Uttar Pradesh, India
Tel: +91-120-2352999
Fax:+91-120-4003007
Email:infostm.india@st.com
CIN:U32109DL1990FTC039906
www.st.com

26th June 2015

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Satyam GUPTA** has undergone internship in our **ICT** Group from **30th June 2014 to 26th June 2015**. He has successfully completed his project on **"Software Configuration Management and Performance Testing ".**

During his internship period, he was found to be sincere and professional in his conduct.

We wish him all the best in his future endeavours.

for **STMicroelectronics Pvt. Ltd.**

Ashish K Chawla
Country HR Director

Acknowledgement

The successful completion of any task would be incomplete without acknowledging the people who made it possible and whose constant guidance and encouragement secured the success.

First of all I wish to acknowledge the benevolence of omnipotent God who gave me strength and courage to overcome all obstacles and showed me the silver lining in the dark clouds. With the profound sense of gratitude and heartiest regard, I express my sincere feelings of indebtedness to my guide **Dr. Rinkle Rani**, Assistant Professor, Computer Science and Engineering Department, Thapar University for her positive attitude, excellent guidance, constant encouragement, keen interest, invaluable cooperation, generous attitude and above all her blessings. She has been a source of inspiration for me.

I am grateful to **Dr. Deepak Garg**, Head of Department, Computer Science and Engineering Department, Thapar University for the motivation and inspiration for the completion of this thesis.

I will be failing in my duty if I do not express my gratitude to **Dr. S. S. Bhatia**, Dean of Academics Affairs in the University, for making provisions of infrastructure.

I wish to place on record my gratitude to **Mr. Deepak Khatri**, Project Manager, ICT, ST Microelectronics Pvt. Ltd, Greater Noida for providing me an opportunity to work with them on this project of such importance. My stay in the organization has been a great learning experience and a curtain raiser to an interesting and rewarding career.

Last but not the least I would like to express my heartfelt thanks to my parents and my friends who with their thought provoking views, veracity and whole hearted cooperation helped me in doing this thesis.

Satyam

Satyam Gupta

(801332022)

Abstract

Now days there are huge evolvement in the field of IT, where people can control and manage their software development according to their need or requirement. The IT organizations need to produce quality product with the most economical and timely efficient way. To achieve this goal there are requirement of various software disciplines collaboration, Software Configuration Management (SCM) is one of the important discipline. With the help of SCM developers can manage different software artifacts. SCM is a tool based approach; SCM tools provide a great interface to reduce developer overhead. Various tools are available in the market. All these tools have their own functionality and capability.

The involvement of Agile Methodology has facilitated developers to manage the product cycle timely and the use of SCM has made the management easier. This thesis explains two case studies; in the first part practical and relevant comparison among three Software Configuration Management tools namely Clearcase, Subversion and GIT has been done. The second part explains the coexistence of Agile and SCM practices which is usually considered as irrational. The research also focuses on the benefits from the user point of view and its role in improving the organizational capability. The thesis also chalks out a plan for successfully implementing this approach.

Table of Contents

List of Figures

List of Table

Chapter 1 Introduction

In general, SCM (Software Configuration Management) controls the evolution and integrity of a product by identifying its elements, controlling and managing change, and verifying, recording, and reporting on configuration information. From the software developer's perspective, SCM facilitates development and change implementation activities [6][18]. It covers the process of determining what to change in software and related documentation.

SCM is a tool based development environment. Various tool are available in market to support this important activity, all these tools have their own capability and features. Use of tools depends on user requirements. Although it is possible to perform Software Configuration Management activities without tools but with the help of these tools we can manage these complex activities very easily and efficiently.

By the SCM we can make all elements which are used in any software development which are manageable and traceable by set of standards and procedures The intention of introducing Agile teams to faster development process by creating cross-functional teams and able to solve all problems related to the assigned functionality development. Like other software development process, agile methodology also requires a team-based development environment, which is supported by the SCM. Unfortunately, some developers considered SCM as an unnecessarily and old controlling discipline. But this is a misapprehension [10].

In the Agile development environment Configuration management process should adapt its work and should support different development teams for ensuring a transparent end-to-end product handling. This is important due to a possibility that in Agile model individual teams implement different new functions on the same software product base concurrently. Agile development methodology is for producing high quality software in least time and efficiently by learning to adapt the changes in programmer specifications, the adapting to changes being commonly regarded as the greatest obstacle to quickly and efficiently producing high quality software [4].

1.1 Background

1.1.1 What is SCM?

SCM is necessary because of high complexity of software and rapid changing nature of software. By using SCM as a strategic tool an organization get an edge over organizations that are not using SCM or using it ineffectively [7]. SCM tools purpose is to provide archival of different versions of evolving artefacts and avoid various development conflicts. A software configuration have existing, functional and physical attributes of a software system as well as combinations of software systems "Software Configuration Management is an umbrella activity that is applied throughout the software process" [9]. We can define SCM as the discipline of controlling the evolution of various complex software systems [12].

SCM is like construction management in which addresses the construction and building of products, process management that ensures the implementation of the organization's procedures, policies and life cycle model, and team work control that deals with the work and interactions between multiple developers [13].

1.1.2 SCM Process variables for SCM Practices [3]

- Configuration Identification
- Configuration Accounting
- Software Change Categorization
- Build Management
- Release Management
- Configuration Status Auditing
- Version and Baseline Management
- Central Change Authority

Table 1.1: Basic SCM Activates [8]

SCM Activates	Description
Configuration identification	Identification of software configuration items. Structured relationship among these configuration items.
Configuration control activity	Monitor and manage change process during SDLC, control, and release process. In this evaluate the change based on cost, impact and others factors.
Configuration status accounting	Creating and organizing reports based on any configuration management data for efficient performance.
Configuration accounting audit	Recording and reporting the status of change requests in the software product and evaluate the accordance of software products and processes to valid regulations, guidelines, standards, plans and procedures

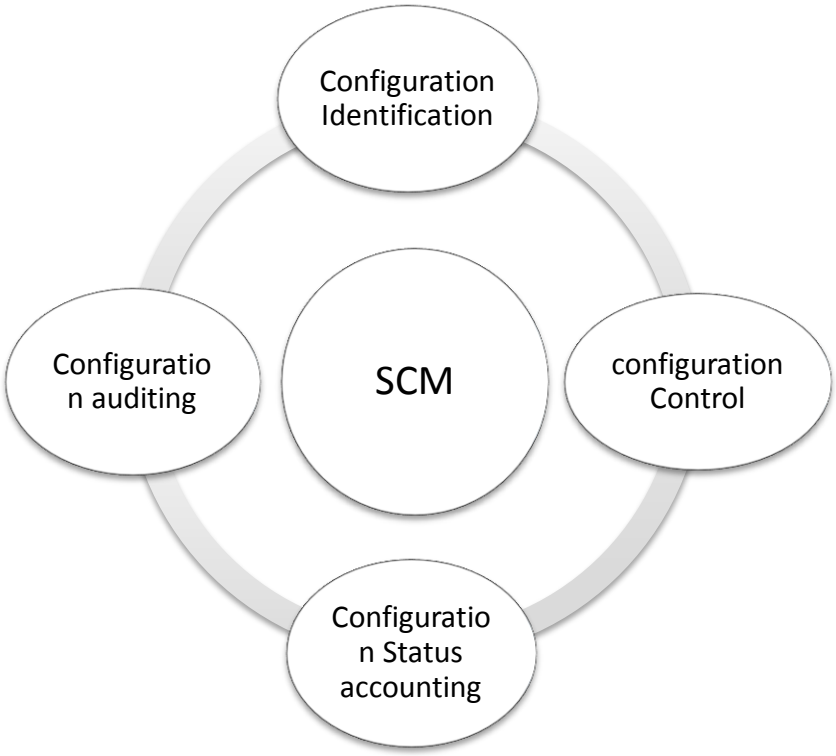

Figure 1.1: SCM Activities

Chapter 2 Introduction to SCM Tools

In any software development process changes occurs in the entire lifecycle. SCM tools provide a way to effectively control changes [14].

2.1 Version Control System

These are tools that help developers to maintain complete version history; this also allows working simultaneously.

There are two types of VCS

- **Centralized version control system:** In this VCS there is centralized server that contains all the history of its version file. When any user wants to work on any item he can set his view and directly works on the repository. This kind approach is very good in terms of audit point of view because all the data and its history at a single place.
  This approach is also having some serious drawback like if central server goes down nobody can perform their operations over that time. A risk factor is also associated with this approach due to single point of failure.
  Clearcase and subversion is example of CVCS.
- **Distributed version control system:** In this repository is fully mirrored over various locations. When any developer wants to work on this system he makes a clone of repository on his local machine. Every checkout provides a full backup. In DVCS user can perform various operations offline.
  GIT is example of DVCS.

There are various tools for the software configuration management and version control system. These tools are foundation for a software development team to collaborate. For our case study we focus on three widely used tools ClearCase, Subversion and GIT. We perform a comparative study of these tools based on practical use.

2.2 ClearCase

ClearCase is one of the famous software configuration management tools which is developed and supported by the Rational Software division of IBM. Various large and medium organizations are using ClearCase to handle their projects with a considerable team size. ClearCase can run support various platforms like Windows, Solaris, and Linux. ClearCase can handle large numbers of files, binary files branching, labeling, and versioning of directories. ClearCase is a key enabler of Automated Software Quality.

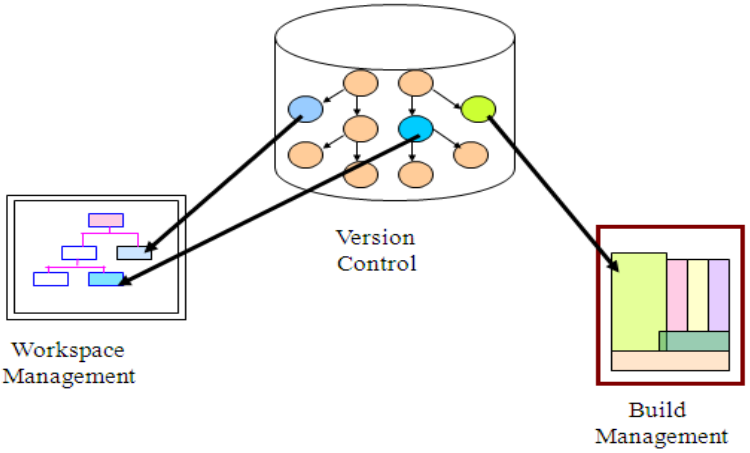

Figure 2.1: Configuration Management

2.2.1 ClearCase Basic Functionalities

- **Version control**: Clearcase can manage files, directories, and other elements throughout software lifecycle.
- **Defect tracking**: ClearCase can manage defect facing by developers during version control.
- **Build auditing**: ClearCase helps streamline the edit-build-debug cycle and reproduce software versions.
- **Parallel development**: ClearCase provides easy branching and merging for the parallel development.
- **Security**: Clearcase offers electronic signatures, user authentication for the controlled access.

2.2.2 Terminology

VOB: A central repository which contain ClearCase elements. VOB repository can reside on:

- on a Windows NT system
- on a Windows 2000 system,
- on a UNIX system

Element: An object that is comprised of a set of versions, organized into a version tree. Examples of elements:

- Source files
- Directories
- Binary files
- Object libraries
- Document

View: A Temporary filtrated user workspace between user and the VOB repository. While working on ClearCase first we have to set a view, the view name should be unique.

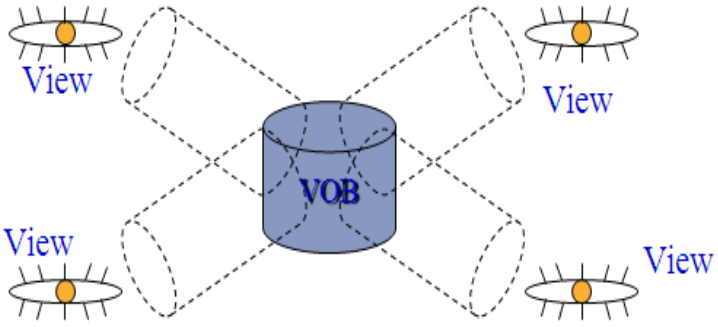

Figure 2.2: Multiple Views

2.2.3 VOB Administration Responsibilities

- **Creating VOBs:** Any user can create a VOB, but the Rational ClearCase administrator is often responsible for creating and managing VOBs that hold artifacts that are important to all members of the community.
- **Importing data:** Rational ClearCase includes a variety of tools that can import data from other configuration management systems. It is often the administrator job to import this data into VOBs and organize it for the use of the community.
- **Backup and recovery:** VOBs are critical data repositories with special backup and recovery requirements. Timely implementation of a reliable backup and restore strategy for VOBs is one of the Rational ClearCase administrator's most important jobs.
- **Managing network access:** Network access information for all VOBs is stored in the Rational ClearCase registry, which is described in administering the Rational ClearCase registry. The registry automatically updated on VOB creation. You do not need to change this information unless you move a VOB, change a VOB server host name, or enable VOB access in mixed networks of computers running different supported operating systems.
- **Managing user access:** For controlling access to VOB data each VOB has an owner, a primary group, an optional supplemental group list, and a protection mode. Understanding and managing VOB access is an important job for ClearCase administrator.
- **Periodic maintenance:** VOB administrators maintain synchronization between the need to preserve important data and the need to conserve disk space. Rational ClearCase collect information regarding data on disk and used this information for occasional cleaning of unnecessary data. It also includes a job scheduling service and tools that you can run periodically to monitor and manage VOB data storage and to check VOB data integrity. Periodic maintenance describes periodic maintenance tasks in detail.
- ClearCase administration also includes routine backup of repository and recovery of lost data.

- Although ClearCase having floating license concept, in which license services provided to user for a particular time. ClearCase admin need to set policy for the license uses.

2.2.4 Challenges in this field

ClearCase is a very powerful tool for SCM, but it faces some challenges

- **Speed**: Even on presence of good network bandwidth dynamic views are slower than local file systems.
- **Sensitivity to network problems**: ClearCase is based on CVCS in which user need to connect with the central server while working. Due to high network utilization it needs high bandwidth. That's by performance of ClearCase depends network.
- **Costly license:** Major problem is that it a licensed version control tool, we have to pay if we want to use. But sometimes it is not possible for small organizations to afford such kind of cost for using it to purchase license is needed.

2.3 Subversion

Subversion is an open source version control tool, which is developed by CollabNet in year 2000. This is a centralized version control system.

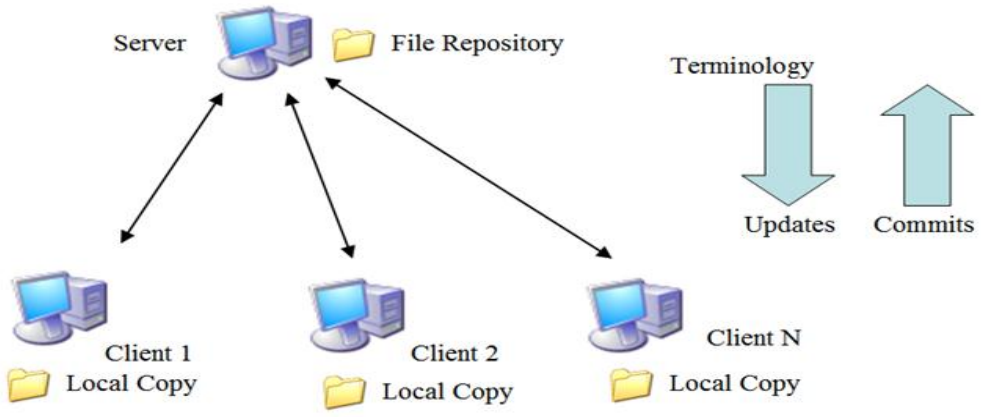

Figure 2.3: Subversion Update and Commit

2.3.1 Subversion's Components

- **svn :** This is a command-line client program
- **svnversion :** This program report the state of a working copy
- **Svnadmin:** This tool is for repairing repository.
- **Svnserve:** Make available repository to others in the network.
- **Svndumpfilter:** This is for filtering repository dump streams.
- **svnlook:** This is for inspecting subversion repository.
- **svnsync :** This is for the synchronizing the one repository to the another in the network.

2.3.2 Features

- **Integration with other tools:** With the integration developer can access SVN from various tools, like eclipse, bug tracker, jira etc
- **Graphical User Interface:** The GUI of SVN supports developers. SVN have the feature of Icon overlay which shows status of working copy.
- **Atomic commits:** With the help of atomic commit developer can add, delete or modify file in one transaction, grouping of operations can be possible.
- **Full Version history:** Repositories contain full version history of file modified, renamed and deleted.
- **Easy branching and merging:** This is flexible in terms of branching, while merging it resolve conflicts. Revert operation can be done at any point of time.
- **Consistent data handling:** In SVN difference with version is shows in both text and binary files. This helps in data hiding.

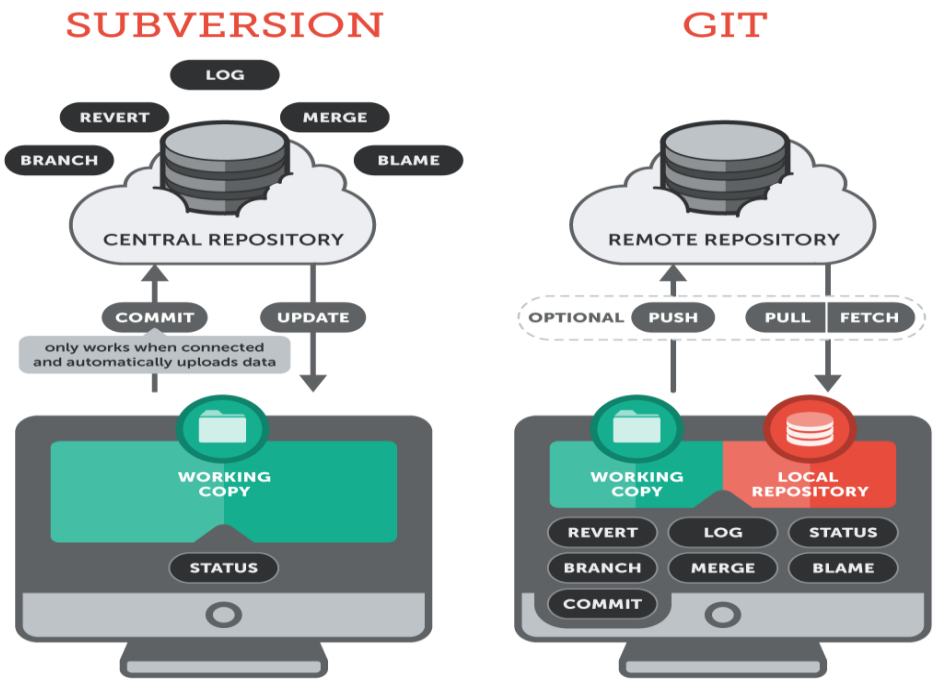

Figure 2.4: Subversion and GIT

2.4 GIT

GIT is a Distributed revision control and configuration management tool. Initially GIT was developed by Linus Torvalds for Linux kernel development in 2005. GIT is free open source software distributed under the terms of the GNU General Public License. GIT comes under DVCS in which every working directory is a replica of remote repository. GIT is not dependent on network access for most of its operations.

2.4.1 GIT Terminologies

- **Repository**: A repository is a complete source database managed under GIT.
- **Clone:** A clone is a copy of a repository. The copy is self-contained with complete history and full revision capabilities.
- **Fork**: From the user point of view, a fork is similar to a clone. A fork can only be performed on the same server. It is mainly a disk space optimization of the clone mechanism.

- **Remote repository**: As GIT is a distributed system; several clones of a repository can exist. Thus referring to a different clone than the local one is known as referring to a remote repository.
- **Origin:** When creating a clone of a remote repository, a link from the local clone to the remote repository is created. In this case, the remote repository is called origin. In the Subversion environment, the origin is the central repository.
- **Upstream**: Within a large development community, there is often a remote repository which is the reference. Most of the time, users are not allowed to commit directly to it. Thus they have to fork this remote repository into a developer area and clone it into their local development area. In the GIT terminology, this remote reference repository is known as upstream.
- **Commit**: A commit corresponds to the storage of a change in the **local** repository.
- **Patch**: A patch is a set of commits representing a (coherent) set of changes.
- **Merge**: A merge action consists in applying modifications from a branch to another one. It has the same meaning as in Subversion terminology.
- **Rebase**: A rebase action consists in un-doing the local commits, updating the current branch to the latest revision of the reference and re-applying local commits. In other words a rebase action applies local commits on top of the latest revision of the reference (in the local area).
- **Push**: A push command synchronizes a remote repository with the local repository, i.e. locally committed changes are applied to the remote repository. Subversion environment, a commit corresponds to the GIT sequence of commit + push.
- **Fetch**: A fetch command retrieves the changes from a remote repository and stores them into a local cache.
- **Pull**: A pull command retrieves the changes from a remote repository and applies them to the local repository (i.e. fetch + merge).

2.4.2 GIT Workflow

There are three important sections of any GIT project.

- **GIT Directory**: This is project metadata and object database storage location.
- **Working Directory**: This is single checkout of one version of the project. This is the place where we modify files.
- **Staging Area**: Simple file which resides on directory that stores info about what will be committed.

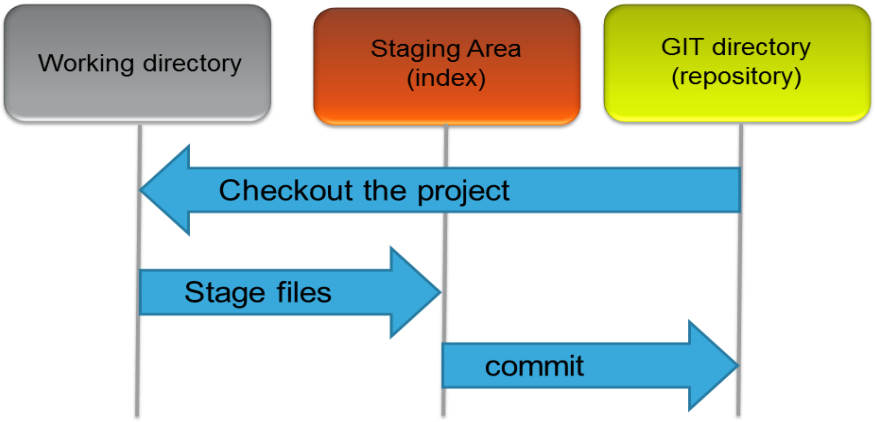

Figure 2.5: GIT Sections

There are three main states in GIT where files can reside:

- **Committed**: Means data is stored in local repository.
- **Modified:** File modified but not committed yet
- **Staged:** File is marked for commit.

1. We make a clone of GIT repository as a working directory.

2. We modify the working directory by adding/editing files.

3. Now we stage the files in staging area.

4. We can review the changes before performing commit operation.

5. If everything is fine we commit changes. And store these changes permanently to GIT repository.

6. Now we push these changes to the repository.

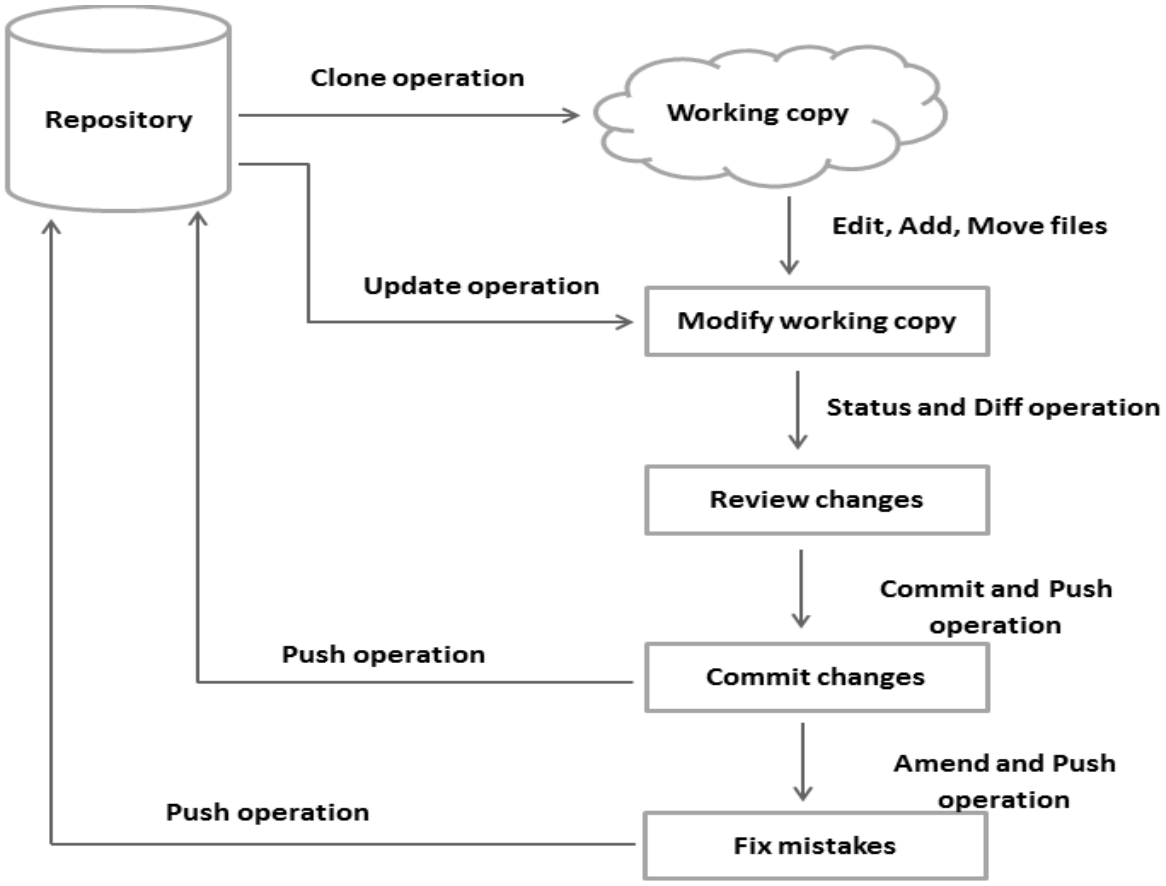

Figure 2.6: GIT Workflow

2.4.3 Working with Remotes: Remote repositories are having the different versions of project than repository that is hosted on the network. We need to push or pull data while interacting with remote repositories.

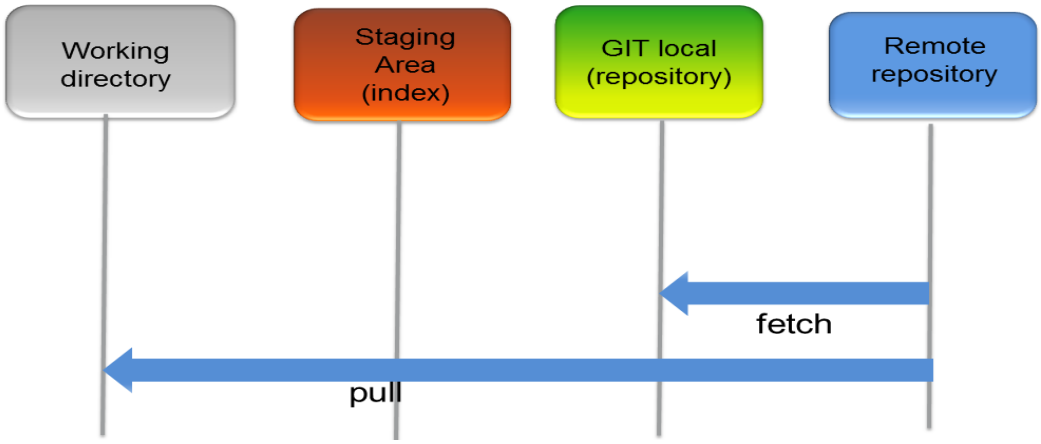

Figure 2.7: GIT Fetch and Pull

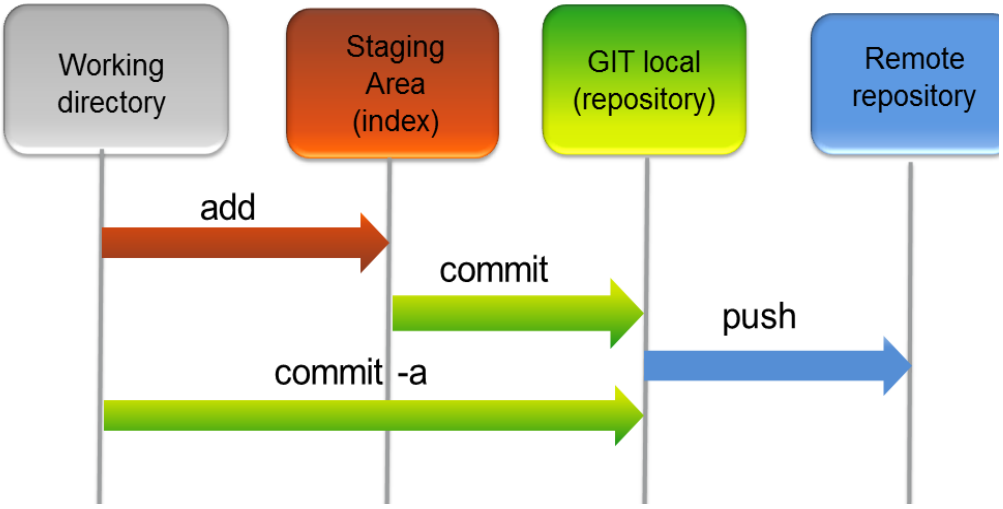

Figure 2.8: GIT Commit and Push

2.4.4 Advantages of GIT

- **Free and open source:** Git releases under open source license. This is freely available on internet. As GIT is open source, we can access to its source code and customized it based on your requirements.
- **Fast and small:** Most of the GIT operations are performed locally, so this provides huge benefit in terms of speed. There is very less dependency on remote or central servers, No need to interact with remote server for every operation.

- **Security:** Git uses cryptographic secure hash function (SHA1). Every file and commit is check-summed by SHA1 and retrieved by its checksum at the time of checkout. So it is not possible to change file, date, and commit message or any other data without knowing GIT.
- **No need of powerful hardware:** In GIT, developers need not to interact with the remote server unless they need to push or pull changes. Most of the operations performed at client side, so the server hardware requirement is very less.
- **Stores Snapshots not differences**: Most of the VCS keep set of files and changes made to files into different versions.

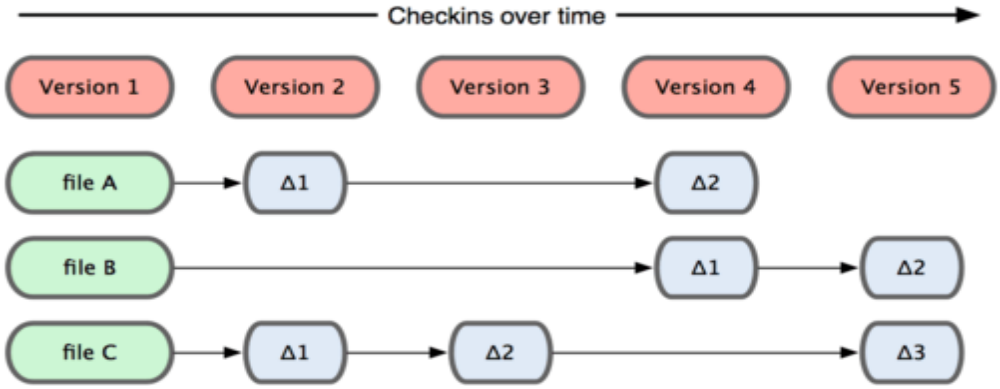

Figure 2.9: How other system store version data

But GIT save state of project every time when commit operation is performed, GIT take snapshot of current state means if any file not have any change then GIT just store the link of previous identical file.

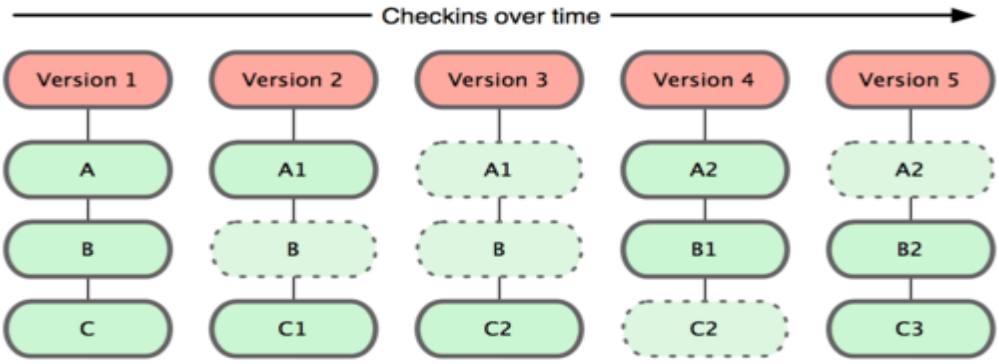

Figure 2.10: How GIT store version data

- **Implicit backup:** There are multiple copies of data over different remote machines, so this is robust in terms of failure.

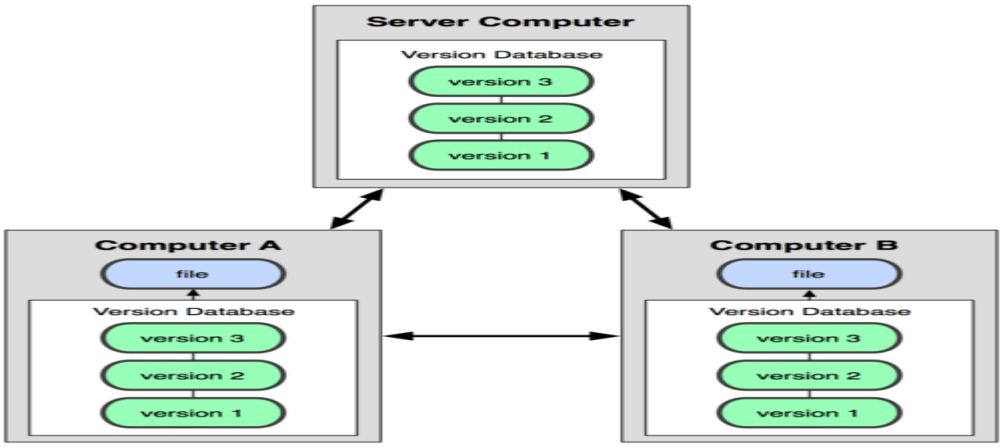

Figure 2.11: GIT synchronization with other systems

Chapter 3 Working on SCM Tools and Comparison

SCM is a tool oriented development environment, where everyone has a choice between poorly integrated, independent tools, or highly integrated, most specialized tools [15]. SCM tools keep record of all software changes history, and merge and integrate various local changes back to the main branch of the code base [20].

3.1 Working with ClearCase

There are two ways of working on ClearCase tool

- **Snapshot view:** In snapshot view user made a copy of repository on local machine, perform operations on that copy then Update the central repository
- **Dynamic view:** In Dynamic view user directly works on repository.

Before working on ClearCase repository user need to create a view. This view could be dynamic or snapshot based on users requirement.

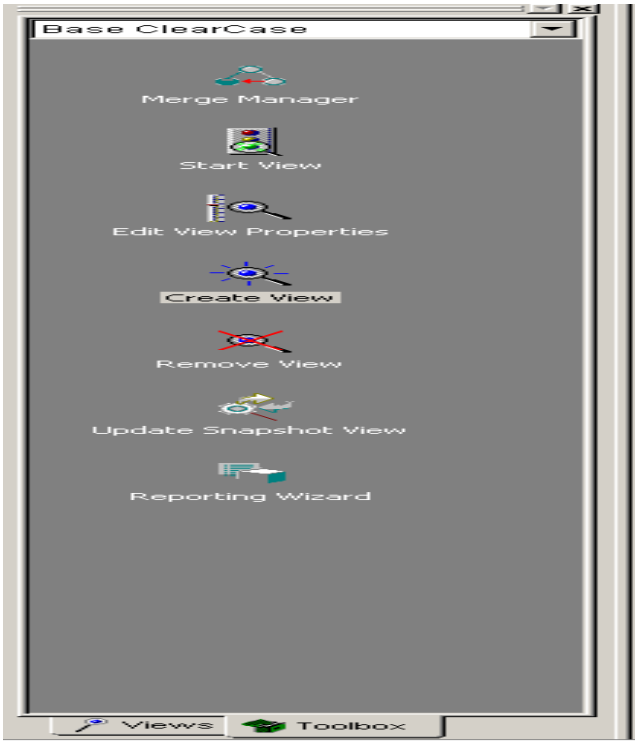

Figure 3.1: Creating View

After selecting the create view option user need to select type of view

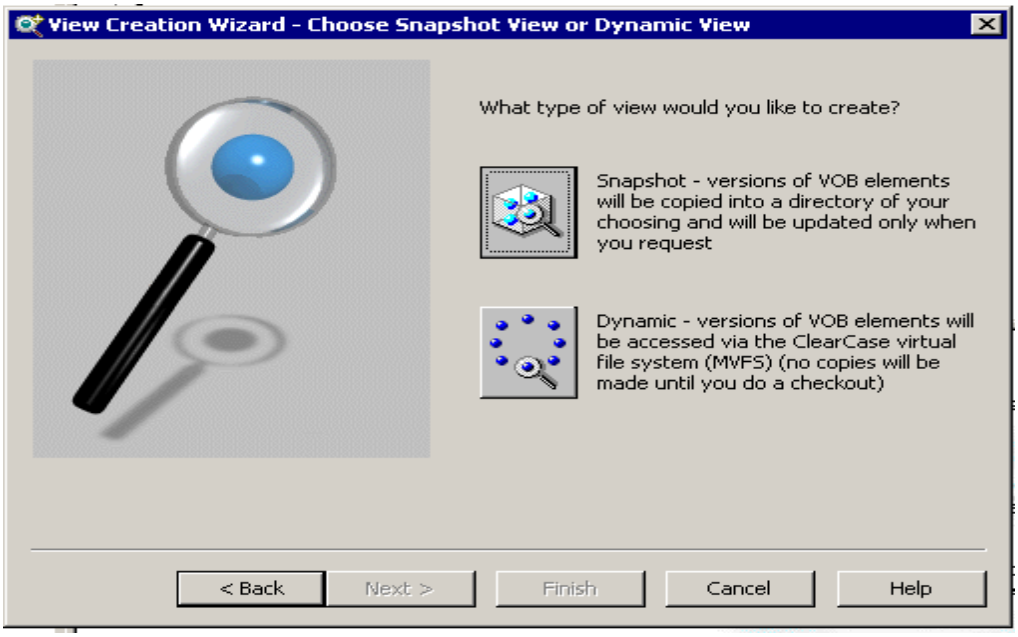

Figure 3.2: Selecting View type

After selecting view type user need to select different repository element which he wants to load on this view.

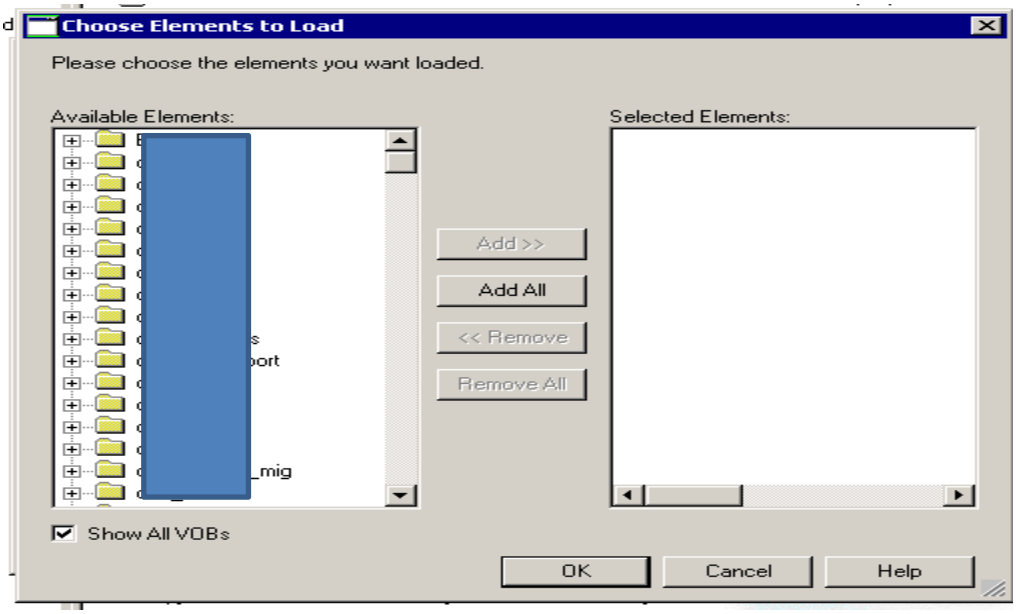

Figure 3.3: Choose Elements to load

- After setting the view set the config specs of that view, which will show load rule over different ClearCase elements. Load rules should be separated by newlines.

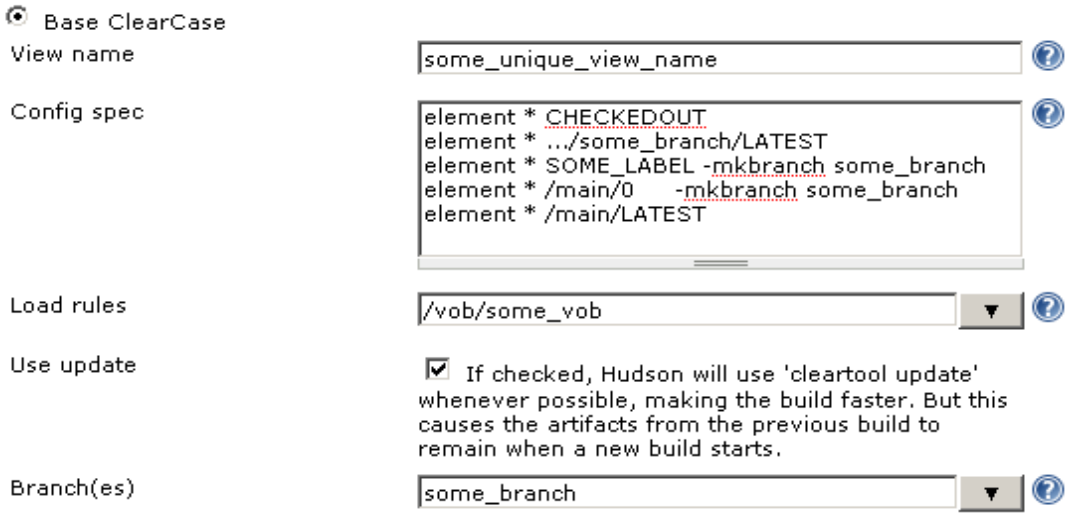

Figure 3.4: Config specs

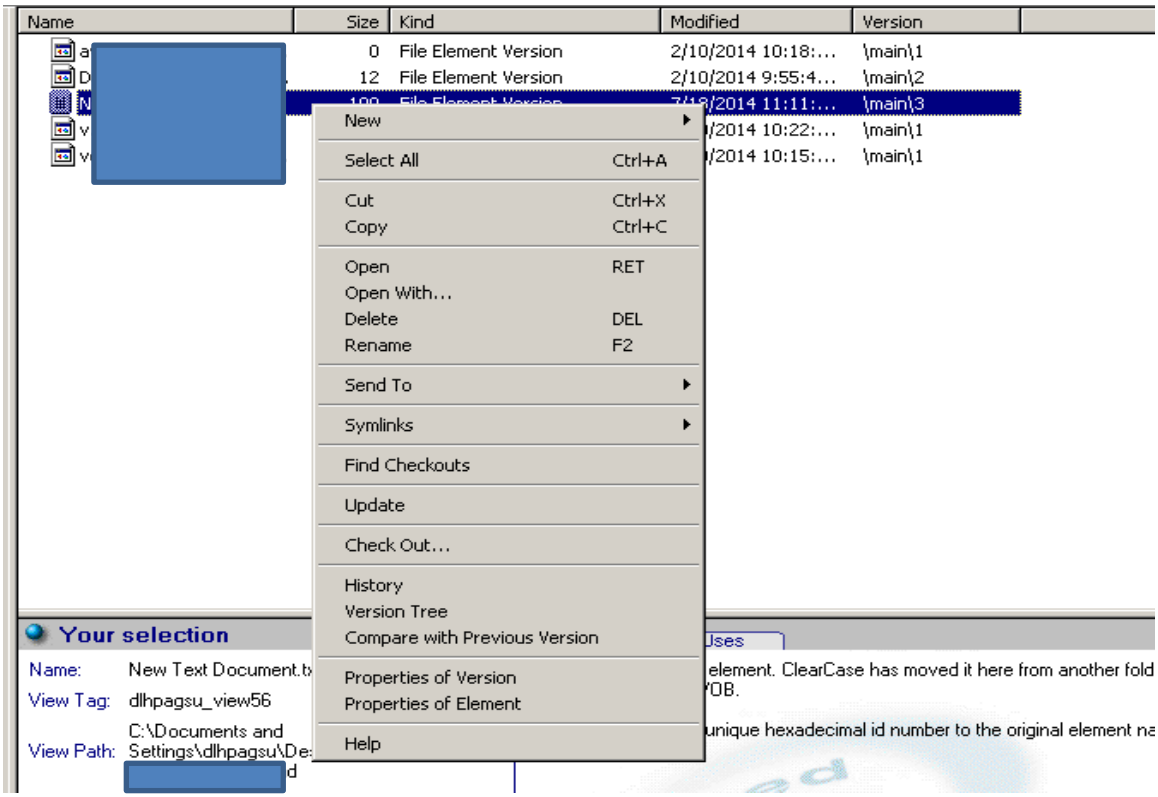

Figure 3.5: Snapshot Workspace

Users work space is ready.

User can perform checkout on any element, perform operation on that then checkout that element.

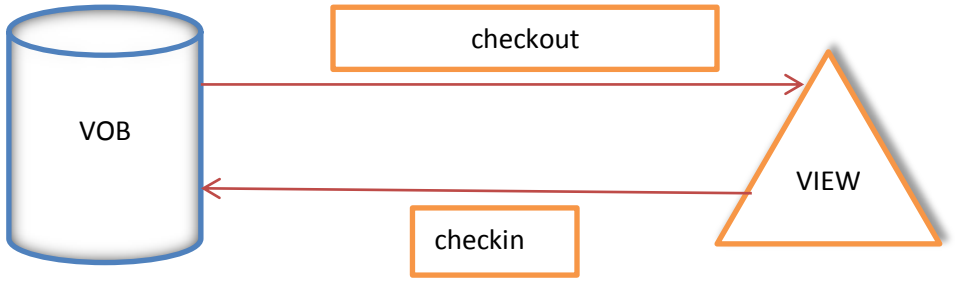

Figure 3.6: Checkout and Checkin

Version tree: this is a pictorial representation of evolution of all the versions of a element

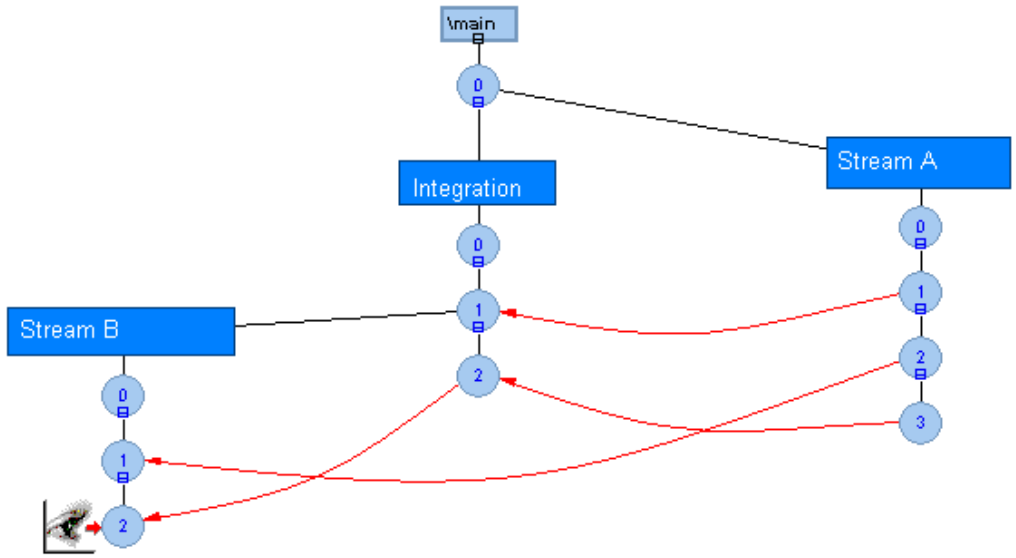

Figure 3.7: Version Tree

3.2 Working with Subversion

Figure 3.8: Subversion workflow

- **Create Repository:** This is a central place where developers can store their data along with its history. This history contains all the changes performed.
- **Checkout:** Before start working on any file we need to create a working copy of that file in our work space. Here developer can perform modification on file.
- **Update:** This operation is used to update latest changes with the repository. As repository is being copied by various workspaces, this command synchronizes the workspaces with the repository.
- **Perform Changes:** Developers can perform various operations over file after taking checkout. These operations are like add, delete or modify existing content.
- **Review Changes:** Before commit changes to the repository developer need to review these changes. The status operation lists all the recent modifications done over that file.
- **Fix Mistakes:** There is option of revert. In case developer not satisfies with the recent changes.

- **Commit Changes:** This is used to apply changes to the repository. After commit other developers can update the recent changes.

3.3 Working with GIT

3.3.1 Design and Implementation

Getting a Repository: There are two approaches for getting a GIT a repository. In the first take an existing directory and import it into GIT. In the second make clone of an existing GIT repository from another server.

Creating a Repository: Before creating repository we need to create a folder under which we want to keep our project. After creating folder right click on folder and choose GIT gui.

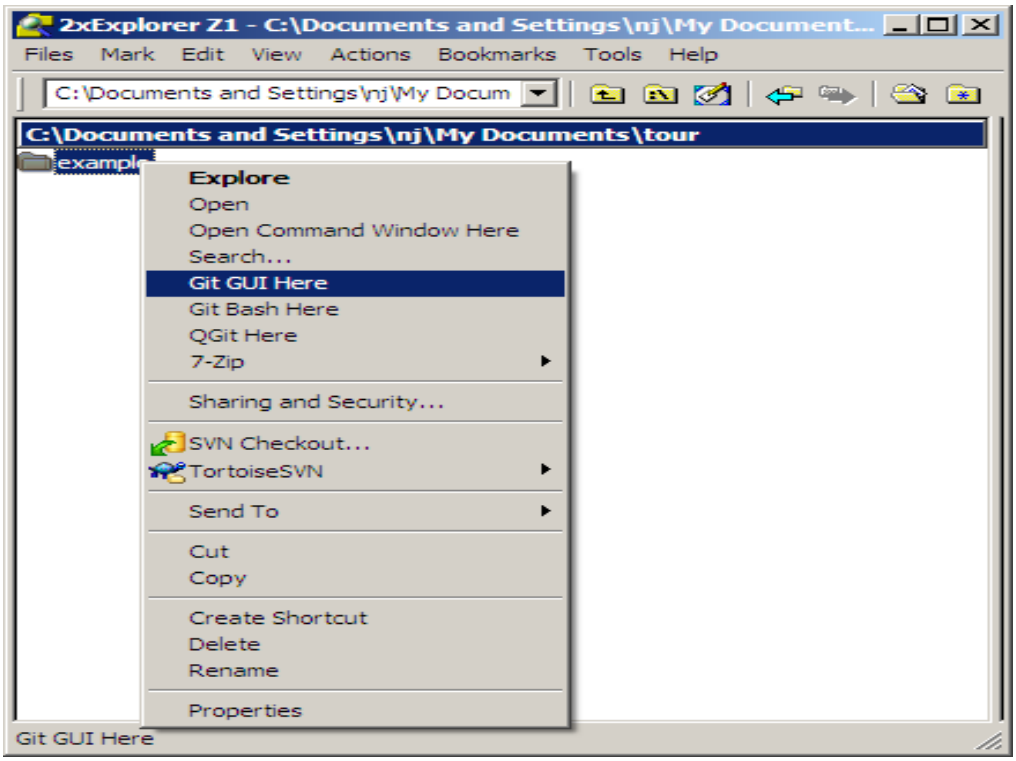

Figure 3.9: Getting GIT Repository

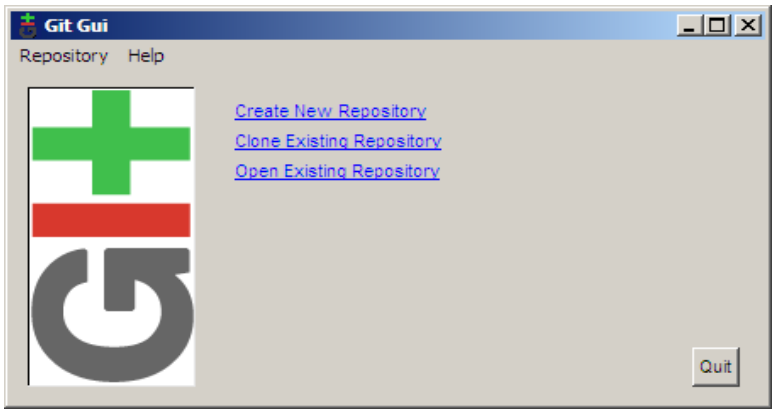

Figure 3.10: Creating GIT Repository

Now choose *Create New Repository*

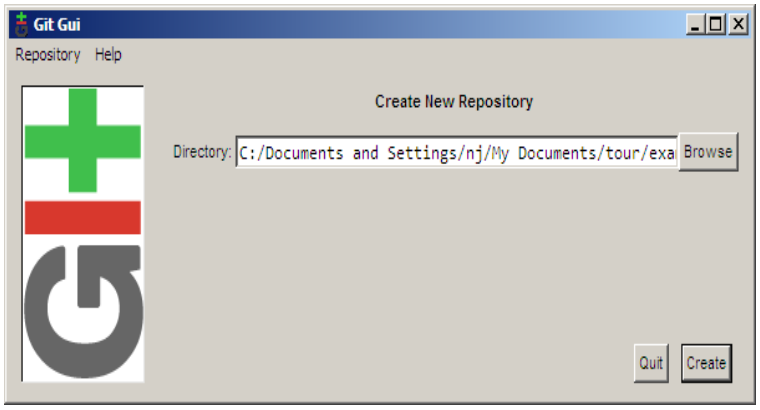

Figure 3.11: Directory location

Fill the complete path to new directory and select Create.

3.4 Comparison between SCM Tools

We analysed these tools (ClearCase, Subversion and GIT) based on different factors. The study of SCM has been concentrated on the use of tools, different SCM models, strategies, techniques [17].

Red	Yellow	Green
Much less	Bit less	Much better

Table 3.1: Tools Comparison

Issue\Tool:	Clear Case	Subversion	GIT
Stability:	In most of the cases it is stable in terms of losing data. If by mistaken user delete data then it shows that data under lost and found category.	This is stable	The product is known to be very fast & Scalable.
Working off-line:	Its snapshot view allows off-line working user only need to be connected for Check in and checkout.	It also allows working off-line user only need to be connected for Commits and Updates.	GIT does nearly all of its operations without needing a network connection, including history viewing, difference viewing and committing.

Recovery from failure, Atomic Commit	No support for atomic commit. Each file need to check in separately.	Supports Atomic Commit. Either every file is committed or nothing is committing	Supports Atomic Commit. Anything which is stored in GIT repository referred by a checksum generated by SHA-1, so this is impossible to change any content without GIT knowing.
Performance	ClearCase creates a heavy load on the serverfor its operations. So this provide low performance	Low overhead on servers for its operations	The performance of GIT is good because most of the operations are not depends on network latency.
Server Administration	Needs ongoing administrative attention	Very low administrative requirements.	Very low administrative requirements.
Backup	For the full backup server needs shutdown.	There are available backup scripts. No need to shut down the server for backup. User can continue its working while backup.	Every repository is a backup of its remote. There are available backup scripts.
OS Support	Linux, Windows, Unix	Linux, Windows, Unix.	Windows, POSIX, OS X, Linux.

Integration with other Software Development tools	Good integration with Windows Explorer, Eclipse and Visual Studio .NET.	Integration of SVN is also possible with Eclipse and Windows Explorer	Integration is possible with TFS and .NET
Incorporation with Bug tracking systems	Possible with ClearQuest	Integration can be possible with tools like Trac or Bugzilla	Good integration exists with tools like Ticgit or BugTracker .NET. We can add tags, save views, comment, manage the state of an issue and change to whom an issue is assigned.
Integration with Microsoft Office applications	Good integration with MS Office applications	No incorporation with MS Office.	Good integration with MS Office Application
Reserved Checkouts	With Reserved Checkout It is possible that one user lock a file and then other users cannot take Checkout until first users release lock by Checks in.	Reserved Checkout is also possible here	With Git, there is no central repository. It has distributed version control system. So no need of reserved checkout.
File merging capability	Merge is possible	Merge is possible	Merging is possible.
History	Users can see history based on	Checkouts history can be seen by	Users can see history of file and directory.

	Checkouts.	users. There is no history of merges between branches.	
Capability of Branch/stream merging	Different branches can be created and possible to merge them.	It is possible to create branches and then to merge between directory trees.	GIT support rapid branching and merging.
User administration	All users who require access to ClearCase has to be member of one of the Groups associated with ClearCase VOB.	While using SVN, individual can use any of the authentication techniques of Apache.	Permissions issue is handling by Gitosis using a single account authenticated by multiple SSH keys.
Access rights	Based on user or group rights are per file or per directory	User can set access rights per directory based while using Apache	Based on user or group rights are per file or per directory.
Remote access without a client	Read access are possible via Web.	when using Apache read access are possible via Web	It is possible to get read access.
Comparing versions (diff)	In Clearcase user can compare text files from the previous version. For the file based comparison there are third party diff tools available.	Here comparison is possible between text files, for the file based comparison there is no special tool.	The good thing with the GIT is its local repository and history storage. In most of the cases it can perform difference operation locally without any remote severs involvement.

Capability to fetch older version with returning	A ClearCase view can point to the previous version based on numbers.	Based on a date or version number it is possible to check-out a previous version.	Without it being noticed by GIT It is not possible to change the previous versions.
Capability to go back to earlier version	User can easily switch between different versions using version tree.	User can easily switch between different versions using version tree.	Using GIT it is possible to go back to previous version.
User interface	Good interface but in Windows Explorer and CC Explorer	Deep integration with Windows Explorer.	Good Interface with Windows Explorer by GITExtension,EGIT for Eclipse.
(Local) Support	Support by IBM.	See local support companies.	See local support companies.
Cost	65K USD per Annum	Free (Open Source)	Free(Open Source)
Views (Workspace)	Dynamic and Snapshot	Only Snapshot	Dynamic and Snapshot
GUI	CC has more mature and complete UI including CLI and GUI	GUI is not that much mature	GIT comes with integrated GUI tools for browsing and committing, but there are a number of 3rd-party tools for users looking for platform-specific experience.

3.5 Results

All the discussed tools have their own importance; The selection of a tool is done based on our requirements.

- Subversion and Clearcase are Centralized Version Control System (CVCS) means they need to depend on network for their operations. But GIT is Distributed Version Control System (DVCS) so network dependency is very low.
- As Subversion and GIT are open source and Clearcase is paid. But on the other side we have a very strong industrial support from IBM for the ClearCase.
- As ClearCase have a large amount of user group, ClearCase have a proven reliability
- The ClearCase is much more User friendly as compare to others.
- ClearCase is much more reliable in terms of data security and reliability, due to this reason many large organizations are going with ClearCase.
- The cost of Clearcase license is very high so this is not possible for small or medium sized organization to pay such a huge cost.
- Clearcase have heavy weight server and client and this required high administration as compare to others.

After analyzing various points related to these tools it is observed that GIT have a huge cost saving as compare to ClearCase. The performance of the GIT is also good due to local repository. The better way for migration could be stepwise migration.

Chapter 4 SCM and Agile

4.1 What is Agile?

Agile is a popular software development methodology. Agile development methodology provides chances to assess the direction throughout the Software Development Life Cycle. This is achieved by regular paces of work, known as iterations or Sprints, at the end of this team must present a potentially deliverable product increment. Most agile development methods used the approach in which there is direct involvement of customers or end users, and the development of functionality in small iterations. Agile methodologies are ideal for projects those have frequent changing requirements, in terms of technology used and capabilities of people[22][23].

A software development methodology which is people centric, communications-oriented, flexible, speedy, lean, responsive, and learning is called Agile methodology [24].

Table 4.1: Basic Agile method characteristics [25]

Characteristics of Agile method	**Description**
Incremental development	Small software releases, with rapid cycles.
Straightforward	The method itself is easy to learn and modify, and it is well documented.
Cooperative	Customer and developers working constantly together with close communication.
Adaptive	Able to take into account last moment changes.

4.2 Need of SCM in Agile

The main purpose of Software development is to produce quality product within described time and budget that satisfy the user requirements. With the help of agile we can perform various activates parallel, and by the SCM we can manage those activities.

Now a day's strong competition between suppliers of IT products and meanwhile technology is also changing, traditional development model not able to deliver a quality product in a short span of time. Due to very short span of time industries are not able to respond to the change properly, this is the main motivation for the Agile with the SCM.

SCM and agile are seems as two non-related entities but they are connected in various ways. With the help of SCM we can improve the quality of software product and processes [2].

Main characteristic of agile is simplicity and speed and the characteristic of configuration management activity is to establish, control and maintain the integrity of a software product configuration throughout a SDLC within feature teams.

Iterative and incremental methodologies like Agile need releases of ready product to support quality testing during continuous development. Business likes to get new product version very rapidly and related IT operations have to support this [21].

4.3 Discussions

4.3.1 In the paper titled "*An empirical study of lean and agile influences in software configuration management*" author has took a web based survey of 158 IT organizations which are already using both the SCM and Agile in their development approach directly or indirectly. The problem statement of the survey was "how does size of organization impact on the use of SCM process in agile software development environment." This study is performed over the different sized organizations. To answer these questions author formed 4 research questions:

- Q1: Does the implementation of SCM practices in the agile software development affected by size of organization?
- Q2: Does all size of organizations appreciated SCM process?
- Q3: Does different software change traceability capabilities having by different sized organization?
- Q4: Does the significance of different management system related with the Software Configuration Management diverge with the organization's size?

With the help of these questions author concluded the result as size of organization does not influence the usage of SCM process in adaptable software development environment. It is also recognized that SCM process consider valuable in different software development environment by all sizes of software organizations.

The major bottleneck of this research is incapability to identify the sufficient samples of the small size organizations which have the SCM process in their agile software development methodology. In the result, this study not able to distinguish the effectiveness of the SCM process for small organizations as compare to medium and large organizations [1].

4.3.2 In the paper titled "*Adaptable software configuration management: An investigation on Australian agile software development organizations*" author's problem statement is "Does the Adaptable Software configuration management process and practices exist in Australian Agile Software Development Organizations?"

To find the analysis result he took the survey of various *A*ustralian IT communities through online. Author take different organizational variables like development team size, organization size, nature of industry and geographical location of organization; he categorized organizations based on their size.

The result of the survey is "There is a variation in the extent of implementing SCM practices within Australian software development organizations having different size". This means that large organizations use SCM more commonly than the medium or small organizations [3].

The outcome of this research work is synchronized with the result of the large sized organizations workshops, Outcome of this workshop stated that Agile methodology facilitated them to become adaptive and flexible to change. Agile practices needed more discipline and tool support. It has been found that all members in this study are using different tools and technologies broadly for software defect, issue management, software build management and release management events.

Although large organizations are using traditional processes such as SCM process in Agile, and average and small software organizations are accepting a tailored form of SCM process for their particular software development requirements.

4.3.3 According to paper "*What, When, Why and How Introducing Software Configuration Management in Agile Projects*" to answer the various questions an interview is conducted with the various project managers of different teams. The study will answer the following questions,

- Do different factors such as experience, group maturity and project development structure influence the introduction of SCM?
- How can Software Configuration Management follow the agile iterative process?
- How can Software Configuration Management be adapted to frequent releases?
- How can Software Configuration Management be adapted to developer feedback?

The result of this interview survey was SCM can be adapted to an agile iterative process without any problems even if they are not explicitly stated in the agile methodologies. Agile methodology often implements SCM unofficially through the tools they use during development.

In the traditional SCM process managing and controlling changes takes a lot of effort and time from development, in contrast to the agile methodology is welcoming changes. A solution is to make the management more oral and try to automate as much as possible.

4.4 How SCM can be implemented in Agile Methodology

4.4.1 In the paper "Impact of agile development implementation on configuration and change management in telecom domain" author proposed a solution for this issue: Software Configuration management activities should be adapted according to the new development process (like Agile). Software Configuration management support should be structured in such a manner that ensure a transparent end-to-end product handling through the development of individual feature team establish control, and maintain the

integrity of a software product configuration throughout a software product life cycle in feature teams.

SCM process modifying framework should be focused on identification of specific process elements and their relationships to project artifacts in new development context with introduced agility. Some of these elements are the following*:*

1. For unique identification of configuration items it is required to have new definition of document numbering for feature related documents and for software product. This means that individual teams can implement different new functions on the same software product baseline at the parallel.
2. In order to control configuration items there should be created product baseline for each feature team. This defines a base where to begin development and where to control changes on the configuration items per each feature team. It is required that there will be feature related baseline and release related baseline.
3. Software Configuration control needs to be recognized within the feature teams, between the feature teams and between feature teams and product responsible manager. Therefore, it will be required to establish CCB (configuration control board) including representatives from all the feature teams.
4. The Change requests will cover the changes related to requirement changes, requirement removal, budget changes, tools update, software configuration changes and implementation changes.

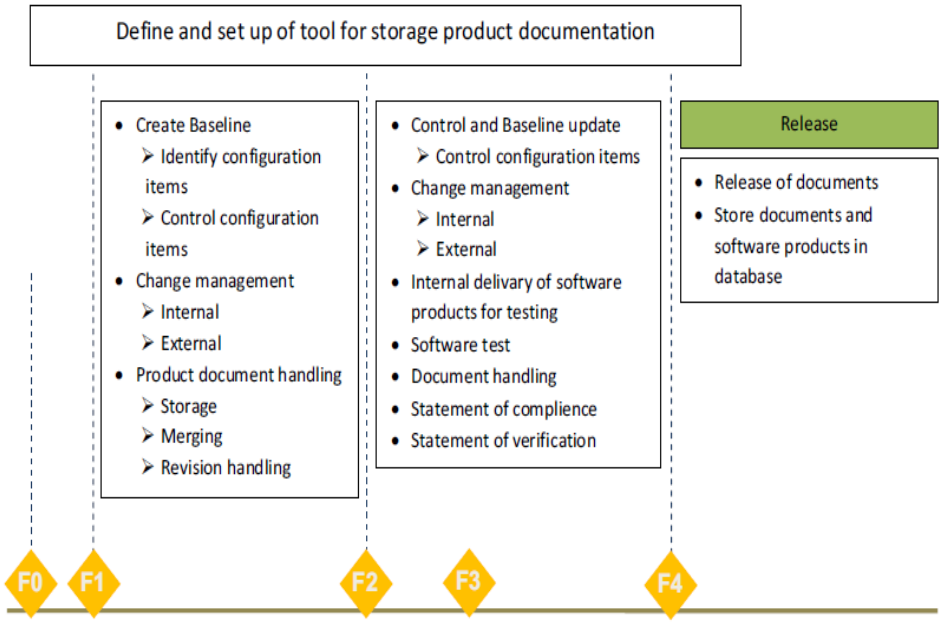

Figure 4.1: Schedule of Configuration activities in new configuration model [2]

4.4.2 Implementing Agile with IBM Clearcase

There is a misconception that enterprise SCM tools like IBM Rational Clearcase cannot support Agile development methodology effectively. However it is not easy to identify configuration elements, and their functionalities, there is a risk of over-engineered SCM process. With the help of tailored SCM many organizations are managing Agile development methods supported by the IBM Rational Clearcase [10].

4.4.3 Implement a Simplified Branching Strategy

Agile projects implements branching strategies. Agile projects encourage continuous integration and refactoring. To achieve this goal there is concept of active development line and release-prep line, in the active development line developer works directly. The release-prep line provides a separate environment for the active release. This release preparation line placed under the development line.

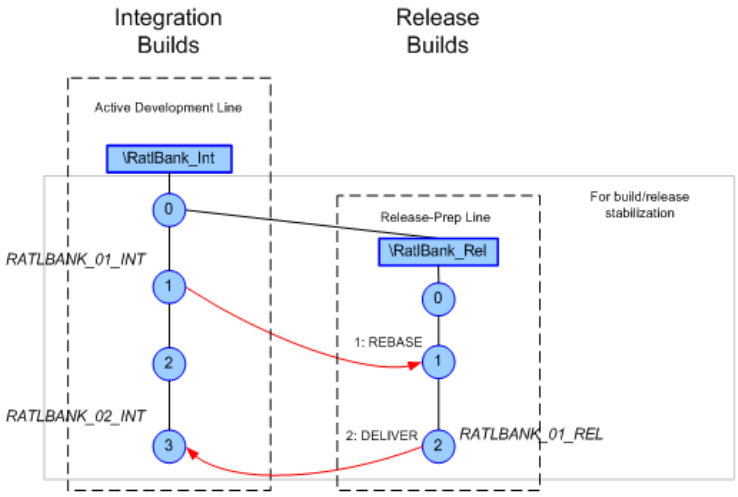

Figure 4.2: Isolate Environment for a Release [10]

4.4.4 Use snapshot-view developer workspaces

In the dynamic views jobs got updated automatically; In the snapshot user need to update every time, here user can update their work space to bring changes to other users therefore this provide user isolation and enough control.

4.4.5 Automate the build process

In continuous integration build process validates newly added code and executes unit tests where required. By this continuous integration user gets knows about integration problem as early as possible. In terms of SCM agile project's build process is set to monitor the integration branch and then execute a build script on check-in [16].

4.4.6 Stage and re-use pre-built binaries

In agile methodology, developers have to deliver and integrate small changes in a very rapid manner; to avoid the long waiting time for a build process this adaptable environment uses concept of pre-built binaries, in this rebuild the whole system only when required. Clearcase used to stage binaries with baseline and label

4.5 Results

Software Configuration Management process is one of the most important pillar of software development process. With the help of agile methodology SCM provides integrate, build, and reconfiguration abilities for rapidly changing software requirements"[3].

Compared with traditional SCM systems, modern adaptable SCM systems provides various functionality like construction, structure, components, auditing, team, process, accounting, and controlling [1].

Theoretically, however, there is no reason why we can't support agile development practices using any SCM tool. Definitely we need not to use all the features of a SCM tool; most of the SCM tools provide some degree of customization. Initially we should identify and define our requirements, which mean understanding development process and how SCM can support it. With the adaptable SCM process for the agile projects organizations can get benefits.

Chapter 5 Conclusion and Future scope

5.1 Conclusion

SCM include both the technical and managerial task. In the current scenario very large amount of elements are involved during software development life cycle, without SCM software development will become improbable task. SCM is a way to manage different software artifacts. Now a day's requirements of users are changing very rapidly, to meet this various processes are being automated. SCM tools are one of the best solutions of this automation. The selection of SCM tools depend on particular organization's requirements.

Agile development methodology sought attention of developers in the last few years. Agile is a way to deliver quality product in a prescribed time frame which leads to customer satisfaction. Software Configuration Management is used to bring control over this rapid changing agile methodology. Currently, there is little significant research on Software Configuration Management in agile methods. The aim of this study was to thoroughly review the existing work with the detailed analysis of various existing open source and commercial tools.

5.2 Future Scope

- This thesis described a practical analysis of three famous SCM tools; this analysis can be further enhanced by taking few latest tools for analysis.
- An integrated framework can be designed to support existing tools using adaptable Software Configuration Management.

References

[1] U. Durrani, Z. Pita, J. Richardson and J. Lenarcic, "An Empirical study of Lean and Agile influences in Software Configuration Management", *In Proceedings of Pacific Asia Conference on Information Systems*, pp. 1-14, 2014.

[2] L. Drvodelic Cvitak and Z.Car, "Impact of Agile development implementation on Configuration and Change Management in Telecom domain", *In Proceedings of the 33rd International Convention MIPRO*, pp. 377-381, May 2010.

[3] U. Durrani, J. Richardson and J. Lenarcic, "Adaptable Software Configuration Management: An investigation on Australian Agile Software Development Organizations", *Lecture Notes on Software Engineering*, Vol.1, No.1 pp. 66-70 ,2013.

[4] J. Estublier, D. Leblang, A.V.D. Hoek, R. Conradi, G. Clemm, W. Tichy and D.W. Weber, "Impact of Software Engineering Research on the Practice of Software Configuration Management", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 14, No.4, pp.383-430,2005.

[5] A. Van der Hoek, "Integrating Configuration Management and Software Deployment", *In Proceedings of the Working Conference on Complex and Dynamic Systems Architecture (CDSA 2001)*, December 2001.

[6] J. Estublier and S. Garcia, "Process model and awareness in SCM" *In Proceedings of the 12th international workshop on Software configuration management*, ACM, pp. 59-74, September 2005.

[7] J. Koskela, "Software Configuration Management in Agile methods", *VTT publications Finland*, 2003.

[8] C. D. Carson, "Relational Metrics Model for Software Configuration Management*", Doctoral dissertation, Southern Polytechnic State University*, 2012.

[9] Keyes, Jessica, "Software Configuration Management", *CRC Press Taylor & Francis Group*, 2004.

[10] http://www.ibm.com/developerworks/rational/library/jun06/lee/.

[11] C. Pardo, F.J. Pino, F. Garcia, M. Piattini and M.T. Baldassarre,."An Ontology for the Harmonization of multiple standards and models." *Computer Standards and Interfaces*, Vol. *34*. No.1, pp. 48-59, January 2012.

[12] W.F. Tichy, "Tools for Software Configuration Management", *In Proceedings of the International Workshop on Software Version and Configuration Control, Teubner Verlag, Grassau, Germany*, pp. 1–20, 1988.

[13] S. Dart, "Concepts in Configuration Management Systems", *In Proceedings of Third International Workshop on Software Configuration Management, Trondheim, Norway*, pp. 1–18, 1991.

[14] M. Kogel, "Towards Software Configuration Management for Unified Models." in *Proceedings of International workshop on Comparison and versioning of software models*, pp. 19-24, 2008.

[15] A. Mahler, and A. Lampen, "An Integrated Toolset for Engineering Software Configurations." *ACM SIGSOFT Software Engineering Notes,* Vol. 13. No. 5, 1989.

[16] www.uml.org.

[17] R. Premraj, A. Tang, N. Linssen, H. Geraats and H.V. Vliet, "To branch or not to branch". In *Proceedings of the 2011 International Conference on Software and Systems Process*, ACM, pp. 81-90, 2011.

[18] http://voer.edu.vn/c/software-configuration-management/edea8e70/9d204001.

[19] H. Volzer , A. MacDonald and B. Atchison, "SubCM: A Tool for improved visibility of Software Change in an Industrial setting", *IEEE Transactions of Software Engineering*, pp. 675-693, October 2004.

[20] Y. Ki and M. Song, " An Open Source-Based Approach to Software Development Infrastructures", *In Proceedings of IEEE International Conference on Automated Software Engineering*, pp. 525-529, 2009.

[21] A. Bartusevics, and L. Novickis, "Models for Implementation of Software Configuration Management", *In Proceeding of ICTE in Regional Development, Valmiera, Latvia*, Vol. *43*, pp. 3-10, December 2015.

[22] T. Chow and D. B. Cao,"A Survey study of Critical success factors in Agile Software projects", *The Journal of Systems and Software*, Vol. *81*, No.6, pp.961-971, 2008

[23] Nerur, Sridhar, R. K. Mahapatra, and G. Mangalaraj. "Challenges of migrating to agile methodologies." *Communications of the ACM*, Vol. 48, No. 5,pp.72-78, 2005.

[24] Qumer, Asif, and B. H. Sellers. "An Evaluation of the degree of Agility in six Agile methods and its Applicability for method Engineering." *Information and software technology*, Vol. 50, No. 4, pp. 280-295, 2008.

[25] http://www.agilemanifesto.org/principles.html

List of Publications

- S. Gupta and R. Rani, "Big Data: An Introduction, Challenges & Analysis using Splunk", *In proceedings of International conference on Advances in Computer Science and Information Technology ACSIT)*, pp. 464-468, April 2015. **[Published]**
- S. Gupta and R. Rani, "Software Configuration Management in Agile Methodology " *IEEE International Conference on Computers, Communications, and Systems 2015*, [**Communicated]**

Video link

[Satyam_Gupta_801332022_Thapar_University]

http://youtu.be/Wp8LHeK_BAw