Empirical Software Engineering (SE-404)

LAB A1-G2

Laboratory Manual

Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY(DTU)

Shahbad Daulatpur, Bawana Road, Delhi-110042

Submitted to: -

Ms. Priya Singh

Submitted by:-

Name: ASHISH KUMAR

Roll number: 2K18/SE/041

INDEX

S.No.	EXPERIMENT	DATE	REMARKS
10.	Perform a comparison of the following data analysis tools. WEKA, KEEL, SPSS, MATLAB, R.	04-01-2022	
1.	Consider any empirical study of your choice (Experiments, Survey Research, Systematic Review, Postmortem analysis and case study). Identify the following components for an empirical study: a. Identify parametric and nonparametric tests b. Identify Independent, dependent and confounding variables c. Is it Within-company and cross-company analysis? d. What type of dataset is used? Proprietary and open-source software	18-01-2022	
2.	Defect detection activities like reviews and testing help in identifying the defects in the artifacts (deliverables). These defects must be classified into various buckets before carrying out the root cause analysis. Following are some of the defect categories: Logical, User interface, Maintainability, and Standards. In the context of the above defect categories, classify the following statements under the defect categories.	25-01-2022	
3.	Consider any prediction model of your choice. a. Analyze the dataset that is given as a input to the prediction model b. Find out the quartiles for the used dataset c. Analyze the performance of a model using various performance metrics.	25-01-2022	
8.	Why is version control important? How many types of version control systems are there? Demonstrate how version control is used in a proper sequence (stepwise).	01-02-2022	
9.	Demonstrate how Git can be used to perform version control?	01-02-2022	

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 8

- ASHISH KUMAR
- 2K18/SE/041

Experiment Objective:- Why is version control important? How many types of version control systems are there? Demonstrate how version control is used in a proper sequence (stepwise).

Introduction:- In software engineering, **version control** (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information.
Version control is a component of software configuration management.
Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. With version control software such as Git, version control is much smoother and easier to implement. Using an online platform like Github to store your files means that you have an online backup of your work, which is beneficial for both you and your collaborators.
For example: If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use. It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, and see who last modified and more.

Version control helps solve these kinds of problems and provides:

- A complete history of every file, which enables you to go back to previous versions to analyze the source of bugs and fix problems in older versions.
- The ability to work on independent streams of changes, which allow you to merge that work back together and verify your changes conflict.
- The ability to trace each change with a message describing the purpose and intent of the change and connect it to project management and bug tracking software.

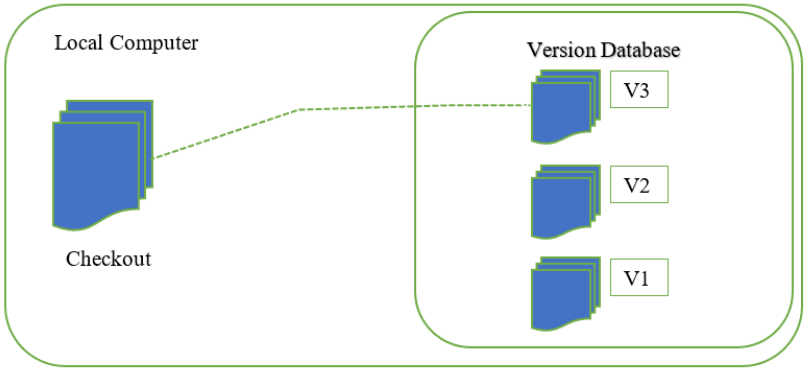
Benefits of the version control system:

a) Enhances the project development speed by providing efficient collaboration,
b) Leverages the productivity, expedite product delivery, and skills of the employees through better communication and assistance,
c) Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
d) Employees or contributor of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,

e) For each different contributor of the project a different working copy is maintained and not merged to the main file unless the working copy is validated. A most popular example is Git, Helix core, Microsoft TFS,
f) Helps in recovery in case of any disaster or contingent situation.

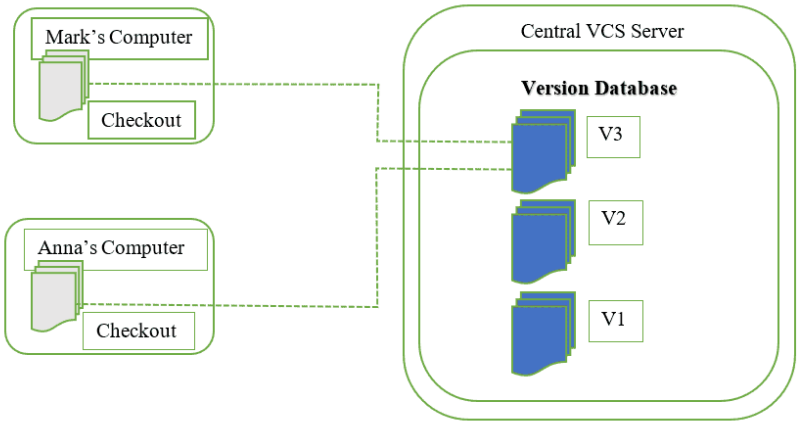
Types of version control Systems:
There are three types of version control: centralized and distributed.

1. **Local Version Control Systems:** It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.



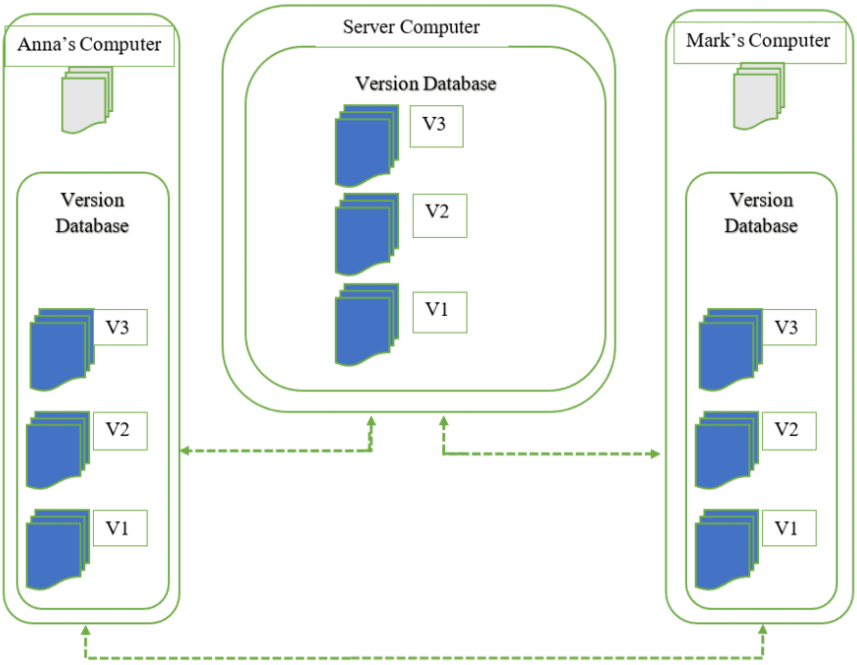
[Fig: Local VCS]

2. **Centralized version control System:** With centralized version control systems, you have a single "central" copy of your project on a server and commit your changes to this central copy. You pull the files that you need, but you never have a full copy of your project locally. Some of the most common version control systems are centralized, including Subversion (SVN) and Perforce.



[Fig: Centralized VCS]

3. **Distributed version control systems**: With distributed version control systems (DVCS), you don't rely on a central server to store all the versions of a project's files. Instead, you clone a copy of a repository locally so that you have the full history of the project. Two common distributed version control systems are Git and Mercurial. With this model, if the server becomes unavailable or dies, any of the client repositories can send a copy of the project's version to any other client or back onto the server when it becomes available. Git is the most well-known example of distributed version control systems.



[Fig: Distributed VCS]

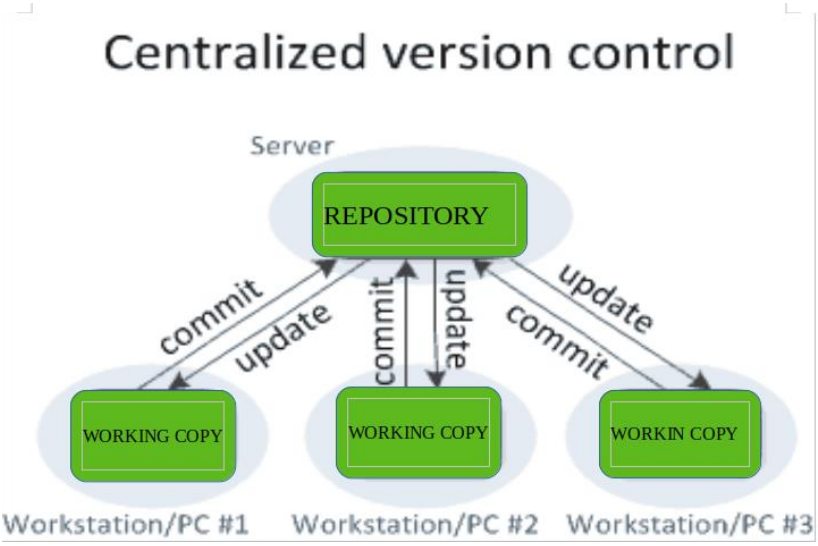
Here are a few of the most popular types of VCS:

- Helix Core (Perforce)
- Git
- SVN (Subversion)
- ClearCase
- Mercurial
- TFS (Team Foundation Server

Note: I have used this source: https://serengetitech.com/tech/introduction-to-git-and-types-of-version-control-systems/

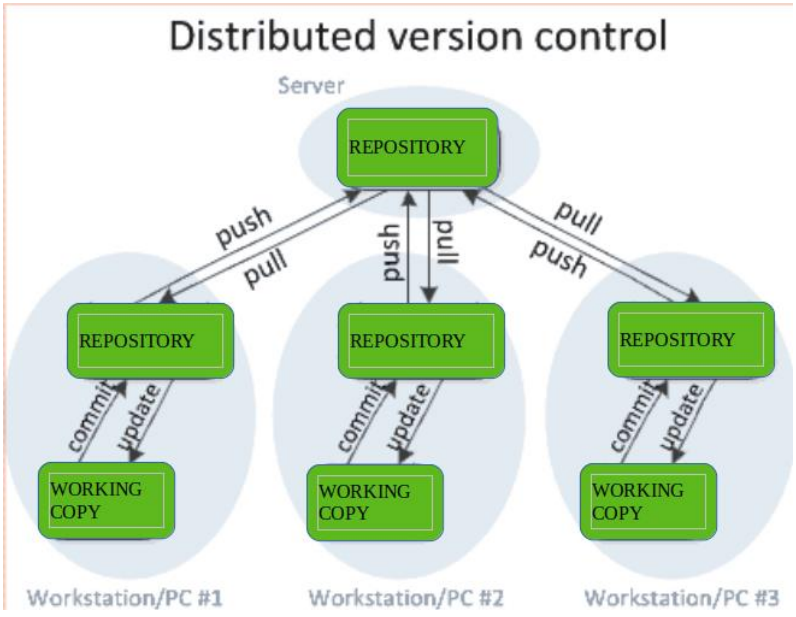
Version control System stepwise proper sequence:

1. Proper stepwise sequence to use VCS in Centralized Version Control System.



[Source: Geeksforgeeks]

2. Proper stepwise sequence to use VCS in Distributed Version Control System.



[Source: Geeksforgeeks]

<u>Learning from experiment:</u>- We have successfully learned about the version control system(VCS) and its benefits. We have also learned about the type of VCS and stepwise sequence of VCS.

Empirical Software Engineering LAB – A1 G2
EXPERIMENT 9

- ASHISH KUMAR
- 2K18/SE/041

Experiment Objective:- Demonstrate how Git can be used to perform version control?

Introduction:- Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. With version control software such as Git, version control is much smoother and easier to implement. Using an online platform like Github to store your files means that you have an online backup of your work, which is beneficial for both you and your collaborators.

Benefits of using GIT as a version control Tool

Having a GitHub repo makes it easy for you to keep track of collaborative and personal projects - all files necessary for certain analyses can be held together and people can add in their code, graphs, etc. as the projects develop. Each file on GitHub has a history, making it easy to explore the changes that occurred to it at different time points. You can review other people's code, add comments to certain lines or the overall document, and suggest changes. For collaborative projects, GitHub allows you to assign tasks to different users, making it clear who is responsible for which part of the analysis. You can also ask certain users to review your code. For personal projects, version control allows you to keep track of your work and easily navigate among the many versions of the files you create, whilst also maintaining an online backup.

GITHUB WORKFLOW:

The GitHub workflow can be summarized by the commit-pull push" mantra.

- **Commit:** Once you've saved your files, you need to commit them - this means the changes you have made to files in your repo will be saved as a version of the repo, and your changes are now ready to go up on GitHub (the online copy of the repository).
- **Pull:** Now, before you send your changes to Github, you need to pull, i.e. make sure you are completely up to date with the latest version of the online version of the files - other people could have been working on them even if you haven't. You should always pull before you start editing and before you push.
- **Push:** Once you are up to date, you can push your changes – at this point in time your local copy and the online copy of the files will be the same.

How to get started?

Step 1: Sign up and installation!

First register on the **Github website**.

If you are on a personal Windows machine, download and install Git for your operating system.

Install Git on Windows:

1. Navigate to the latest Git for Windows installer and download the latest version.
2. Once the installer has started, follow the instructions as provided in the **Git Setup** wizard screen until the installation is complete.
3. Open the windows command prompt (or **Git Bash** if you selected not to use the standard Git Windows Command Prompt during the Git installation).
4. Type git version to verify Git was installed.

- If git is successfully installed in your computer, then you will see something like this when you type this command in cmd:
  git
  git –version

Now you're ready to start using Git on your computer!

To get started, you can create a new repository on the GitHub website or perform a git init tocreate a new repository from your project directory.

From the terminal

Here's how you can get started right from the terminal:

- If you have a project directory, just go to your terminal and in your project directory runthe command:

  git init

- If you want to initialize your project with all of the files in your project directory, runfollowing command to include everything

  git init .

- Let's say you have a folder for your project called "new_project." You could head on overto that folder in your terminal window and add a local repository to it by running:

  cd
  new_project
  git init

- Now you can add files to the staging area one by one with:

  git add<filename>

  or run

  git add .

- to add all of your files to the staging area. You can commit these changes with thecommand:

  git commit –m "<add a commit message here>"

- Now, copy the remote repository URL provided by github to you when you published your repository on GitHub.

  git remote add origin https://github.com/yourusername/your-repo-name.git

- In the last step, use the below command line in your terminal to push the local repository to GitHub. It will upload the file or project on github.

  git push origin master

- You can check whether or not you have changes to push through any time by running

  git status

- Pull the desired branch from the upstream repository. This method will retain the commit history without modification.

  git pull origin master

That's it! You can now initialize a repository, commit files, commit changes, and push themthrough to the master branch.

Learning from experiment:- In this Experiment we learned about how to work with git and understand its workflow.