Proficiency in Distributed Version Control System

The concept of managing subsequent versions of source code of a software project, and any other related document, has been around since the dawn of software development[4]. The idea of a version control system was first introduced in 1972 by RochKind[5], to manage changes in files during development, it was heavily file focused, used by unix, did not have network accessibility and locking-based.

The next generation was centralized and designed for collaborative development. It allowed simultaneous modification, where the user merged current revisions into their work before commiting. It used the client-server architecture to enable collaboration between developers working in a team.

Distributed Version Control System (DVCS) which is the most recent evolution of version control systems became popular because of its ability to permit users to create multiple instances of that repository, supporting decentralization and also kept track of changes in the repository tree using the concept of the directed acyclic graph.

CLASSIFICATION OF SKILL

Distributed Version control systems have the properties of generic soft skills because it is largely independent of software development and is still relevant for other disciplines too but a lack of the understanding of a version control system in software development will hinder the ability to keep track of changes in the artifacts produced during collaborative development of a software project. This supports the fact it has the property of context-sensitive soft skill.[2]

Prerequisites for Skill

Due to its simplicity and relatively shallow learning curve, there is no major prerequisite a new user has to have other than the basic understanding of the file system which is two dimensional(consisting of file and directories).[7]

The knowledge of the concept of directed acyclic graphs will also give a little advantage when understanding the model of storage which DVCS implements.

RELATED SOFTWARE ENGINEERING AREA

There is a possibility that the management involving the content of a software project can quickly go out of control. Software configuration management ensures this does not happen by improving the reliability of information and also managing change within the artifacts of the Software. It has two components namely: Version control and issue management[4].

Version control offers traceability by keeping a complete picture of what changed, who performed it and the reason for the change.

Rationale for Skill

Tracking changes during software development requires that the collaborating team members have a workflow when committing changes to the central repository to avoid merge conflicts. The DVCS offers different model of workflow to manage collaboration, some are heavily focused on branching patterns, and others are repository oriented.[6]

In Devops, DVCS ensures that dependency issues encountered by modern containerized applications are avoided. This is achieved because DVCS offers traceability and visibility into changes made to code .

Roles for Skill

There are several roles in the software domain today. Some of the roles include:

I. Datascience:
II. Software Engineering
III. Devops
iv. Machine learning

All of which requires a form of version control to keep track of changes in the artifacts produced.

Work Related to Skill

The repository contains a set of versions which keeps records of commits by every developer involved in a software project. Information and metrics can be retrieved from the repository. DVCS produce artifacts that can be used during activities in software

reverse engineering. The artifact could help the measurement team evaluate the evolution of maintenance during the lifecycle of a software.

Real-World Example/Scenario of Skill
https://www.youtube.com/watch?v=2sjqTHE0zok&t=2s

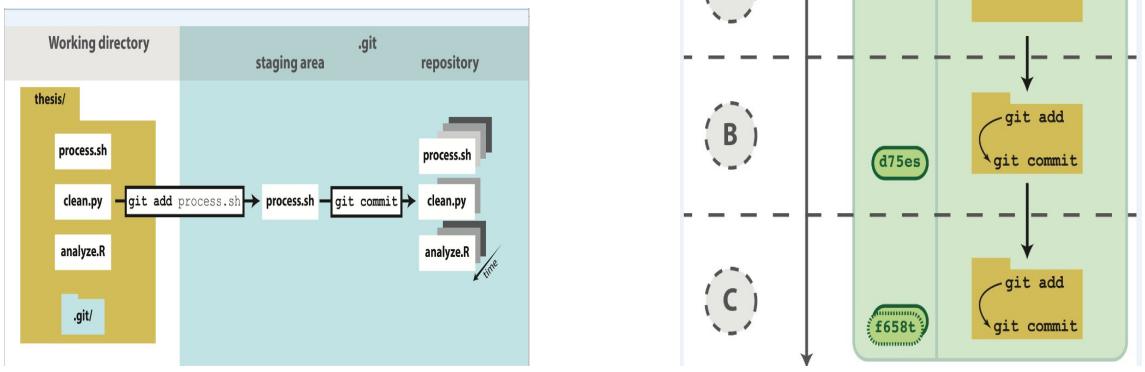

Fig 1.

Role of Academia or Industry in Cultivating the Skill
Course projects which require some form of documentation or adhering to any of the available software development life cycle has ensured that students have a knowledge of distributed version control systems in order to collaborate and complete different tasks required in completing the project.
IT companies have gone ahead to introduce different types of workflow with DVCS to ensure continuous integration[6] thereby ensuring the skill never goes out of style.

Tools Supporting the Skill
There are various tools today implementing concept of distributed version control system[5] that have different ways to store versions of a file and they include;
Mercurial, Git, Bazaar and Veracity.

Bazaar and Git use the Snapshot model in which each new version of the file is independent of its previous parent version and the access time for any of the stored versions of the file is asymptotically constant. Delta based storage model is usually applied by Mercurial.

Veracity adds database commits as another artifact to the set of contents which it tracks.

Skill Self-Assessment
The comprehensive capability to act appropriately and resolve issues that may arise from merge conflicts which may get complex when a workflow is not implemented while using Git makes me competent enough. On a scale of 1 - 10, competency will be 7.[7]

REFERENCES

1. Andy Robinson and Nishant Saini, *Configuration Management and Version Control*, Mar 2017. Available:https://www.automationworld.com/home/blog/13316896/configuration-management-and-version-control.
2. D. Knittl-Frank, "Analysis and comparison of distributed version control systems," Master's thesis, University of Applied Sciences, Upper Austria, jun. 2010.
3. Erick Sink“Version Control by Example”. July 2011. Available:https://ericsink.com/vcbe/vcbe_usletter_lo.pdf
4. Giacomo Ghezzi, Michael Wursch, Emanuel Giger, and Harald Gall: An architectural blueprint for a pluggable version control system for software(evolution) analysis, Association for Computer Machinery. 2012.
5. M. Rochkind, "The source code control system", *IEEE Transactions on Software Engineering*, vol. -1, no. 4, pp. 364-370, 1975. Available: 10.1109/tse.1975.6312866.
6. Chacon and Straub Pro Git. Second Edition. Apress. 2014.
7. Sedelmaier, Y. and Landes, D., 2015. SWEBOS – The Software Engineering Body of Skills. *International Journal of Engineering Pedagogy (iJEP)*, 5(1), p.20.