Edgar Arroyo

January 25, 2025

Module 3.2 Assignment - Version Control Guidelines

Version Control Software (VCS) is a key tool for helping development teams collaborate effectively, track changes, and revert to previous versions when necessary. As development methods and tools have evolved, so have the guidelines for using version control effectively. This paper explores version control guidelines from a variety of online articles, which highlight best practices for using VCS.

Two articles I decided to feature in this paper are from 2024. The other is from 2020. Despite the differences in the software development landscape and VCS environments at the time they were written, all three sources share many of the same best practices, such as:

* Commit Frequently and Meaningfully: Committing frequently prevents work from being lost and makes it easier to identify and resolve any issues.
* Develop Effective Branching Strategies: Use established branching models such as Git Flow, GitHub Flow, or GitLab Flow.
* Write Descriptive Commit Messages: A good commit message explains both the “what” and the “why” behind the change, making is easier for other team members to understand.
* Conduct Code Reviews Before Merging: Performing peer review and testing changes before merging ensures code quality and reduces the likelihood of issues arising later.

A big difference between the article from 2020 and the ones from 2024 is that the article from 2020 encourages committing only complete builds. The 2020 article warns that “an incomplete commit of an API, for example, might build locally, but could break in another team member’s work area.” In contrast, the articles from 2024 encourage “small, atomic commits”. The article on [medium.com](http://medium.com) explains, “this practice makes it easier to track changes, review code, and debug issues.” This difference highlights the shift in modern software development toward Continuous Integration/ Continuous Deployment (CI/CD) pipelines, where smaller, incremental changes are preferred.

The best practices I would add to my list as “most important” are all of the shared practices I listed above. These guidelines are fundamental to maintaining a smooth and efficient version control system. Regular and meaningful commits ensure changes are well documented and prevent loss of work. Effective branching strategies enable teams to manage their workflows and avoid conflicts during development. Descriptive commit messages enable clear communication within the team, making it easier for everyone to understand the purpose of each change. And conducting code reviews before merging ensures that any errors are caught early, ensuring code quality. In addition, I would add that proper integration of a CI/CD pipeline is a crucial element of version control. This helps teams verify that new changes won’t break the existing system.

Sources:

Eng, Adnane Arharbi. “Git Strategies: Best Practices for Effective Version Control.” Medium, 19 May 2024, [medium.com/@aa.adnane/git-strategies-best-practices-for-effective-version-](http://medium.com/@aa.adnane/git-strategies-best-practices-for-effective-version-) control-2a420d7a6776.

INSiGENe. “Version Control Is an Indispensable Aspect of Reproducible Analytics, Ensuring That Every Change to Your Codebase Is Tracked and Documented. This Article Delves into the Essentials of Version Control with 𝙶𝚒𝚝 and 𝙶𝚒𝚝𝙷𝚞𝚋, Highlighting Best Practices to Manage Your Projects Effectively and Co.” Linkedin.com, 28 Aug. 2024, www.linkedin.com/pulse/version-control-best-practices-git-github-insigene-1ahrc.

‌

Schiestl, Brent. “8 Version Control Best Practices.” Perforce Software, 21 May 2020, www.perforce.com/blog/vcs/8-version-control-best-practices.

‌