Jonah Aney 11/09/25 CSD380 Module-3.2 Discussion: Assignment: Version Control Guidelines

In our current professional world, we can access information and stay connected wherever we are in the world. Companies today have teams disbursed, whether that's across a building, campus, region, or globe. With these new capabilities comes some complexities as well. If a team of 8 is working on a project in various locations, who was the last to work on it? Where is the latest version? Has this element been added yet?

Version control allows teams to stay connected, aligned and efficient on their projects. It's an organizational system that stores files/doscuments and enables users to track changes and progress. This gives teams the ability to collaborate and develop a project efficiently, avoiding duplicating or overriding work. The following report will discuss the importance of version control guidelines by reviewing three sources of valuable information. I'll discuss and compare their best practices, see how they relate to today, and conclude with my own list of quality guidelines to follow.

Our first source, DocuWare - "The Ultimate Guide to Document Version Control", discusses how version control helps teams manage, share, and review documents. Not only does it help prevent data loss, it also aids in security and audits by providing a history of the edits, ensuring comliance and limiting overall confusion. The article mentions the following points as a few best practices.

- Use clear naming conventions for identifying files and their versions.
- Restrict editing right to only authorized users.
- Maintain traceability and retore points to reference and understand project history.

The second source, Daily.dev - "Documentation Version Control Best Practices (2024)" states the documentation process should be approached with the same discipline as the program code, and encourages automation and teamwork. A few key guidelines from this site are:

- Integreate automation and utilize CI/CD pipelines for builds, backups, and documentation.
- Keep documentation aligned with product updates so information is current and consistent.
- Discuss with department and teams a plan for version control so the mission and goal is known and implemented effectively.
- Write clear commit messages about what was updated.
- Peer review documentation before merging updates.

Finally, the third source, Git Lab - "Version Control Best Practices" talks about similar points empahzing CI/CD pipeline, a structured process with automation and organization.

Jonah Aney 11/09/25 CSD380 Module-3.2 Discussion: Assignment: Version Control Guidelines

- Develop on feature branches so the main branch remains stable.
- Make small commit updates to keep updates simple. Use descriptive words, like “update”, “fixed”, “added”.
- Ensue code reviews are done before merging branches.
- Use tags and semantic versioning to track changes and automate workflows.

Below is a table quickly comparing the three sources and their approach to the commons facets of version control.

Aspect	DocuWare	Daily.dev	GitLab
Focus	Business & document management	Documentation management	Software development
Goal	Accuracy, traceability, compliance	Collaboration, automation, doc quality	Continuous integration, reliable code
Branching	Minimal / manual version naming	Recommended for doc updates	Essential (GitFlow, feature branches)
Commit Practice	Emphasizes version naming	Clear messages for docs	Small atomic commits
Automation	Optional	Encouraged	Required (CI/CD, tests)
Security	Access control	Permissions & backups	Protected branches, secret scanning
Collaboration	Shared document editing	Peer review	Merge requests & code review

Note. Table created and adapted from DocuWare (2024), Daily.dev (2024), and GitLab (n.d.) with assistance from OpenAI’s ChatGPT (2025).

When reviewing this information, a couple points might of been more useful in the recent past, but are now less relevant since technology progresses so quickly. While it’s still good to maintain a local backup, cloud backups are done automatically to ensure data is not lost making local back ups somewhat less relevant. Version naming was also

done manually in the past but Git and Mercurial track and automatically timestamp each commit with an identifier. This makes manual naming redundant and unnecessary.

This is a short list of the key guidelines I think are most effective:

- Commit changes often. Small commits eliminate confusion and keep progress moving.
- Communicate clear commit update massages using descriptive words.
- Establish and require peer reviews before merging.
- Protect the main branch by only allowing approved changes to merge. Only authorized users have access to make the changes.
- Automate testing and validation with CI/CD pipelines.
- Keep product changes aligned with documentation so information is consistent.

These guideline are valuable because they reflect modern, practical practices for development and workflow. Frequent commits and CI/CD automation for efficiency, reviews and access control for quality and security, and versioning all files for consistency. These guidlines apply to all aspects of the DevOps environment including software, infrastructure, or documentation.

Sources:

Daily.dev. (2024, February 15). *Documentation version control best practices (2024).* https://daily.dev/blog/documentation-version-control-best-practices-2024

DocuWare. (2024, May 30). *What is version control & why is it important.* DocuWare Blog. https://start.docuware.com/blog/document-management/what-is-version-control-why-is-it-important

GitLab. (n.d.). *Version control best practices.* GitLab. https://about.gitlab.com/topics/version-control/version-control-best-practices/