David Amos

Module 3 Assignment

1/24/2025

Source 1: GitLab. (2025). *What are git version control best practices?* https://about.gitlab.com/topics/version-control/version-control-best-practices/

Guidelines:

* Make small incremental changes
* Keep commits atomic
* Develop using branches
* Write descriptive commit messages
* Obtain feedback through code reviews
* Identify a branching strategy

Source 2: Schiestl, B. (2020, May 21). *8 version control best practices*. Perforce Software. https://www.perforce.com/blog/vcs/8-version-control-best-practices

Guidelines:

* Commit Changes Atomically
* Commit Files With a Single Purpose — Not as a Backup
* Write Good Commit Messages
* Don’t Break Builds
* Do Reviews Before Committing to a Shared Repository
* Make Sure Every Commit Is Traceable
* Follow Branching Best Practices
* Protect Your Assets

Source 3: Kramer, N. (2024, July 15). *Documentation version control: Best practices 2024*. Daily.Dev. https://daily.dev/blog/documentation-version-control-best-practices-2024

Guidelines:

* Set up a clear version control plan
* Make good commits with clear messages
* Use branches for different features or releases
* Implement regular reviews and teamwork
* Keep documentation in sync with code changes
* Use continuous integration for automated updates
* Implement security measures and access controls
* Regularly back up and test recovery processes

All articles stressed the importance of creating clear meaningful commit messages. This makes it clear what the purpose of that change was. The first two articles recommend making atomic commits meaning each commit is specific to one task. If you have two tasks to do, then that should be two commits. This makes code reviews easier and clearer and keeps versions and descriptions precise. Using branches is another point that all three articles touch on. Using branches keeps versions of code separated until they have gone through proper approvals to merge into the main branch.

The first article mentioned identifying a branching strategy. I think that is an important step. There are different types of branching strategies such as stable main or trunk based which determines at what point code is merged into main. I think it is important to be consistent and determine which way will be used. A CI/CD pipeline is a great way to enforce consistency.

The third article covers more on the topic of documentation. I think documentation is also an important piece of version control. It can be frustrating to find out a README file or other documentation is not up to date. Documentation is important for identifying bugs, helping future programmers, and for audit purposes.

The second article mentions making sure every commit is traceable. While that is important it seems less important to have as a guideline. Today most version control systems enforce login to track the author of the commit.

Due to these points mentioned above my list of the most important guidelines would be:

* Create meaningful commit messages
* Make atomic commits
* Develop in branches
* Identify a branching strategy
* Keep documentation up to date with changes