Tatiana Tan

Assignment 3

Due: April 6th, 2025

Version Control Guidelines

Version control systems are used by developers to help keep track of the changes that are made within a code over a period of time. It is a super important practice used in software development by allowing teams to work together and collaborate in a more effective way. As the world of technology is evolving, so have the various guidelines used in version control systems. The guidelines for these VCS allow software development teams to avoid common mistakes and streamline their workflow. This paper will explore three different guidelines from different sources and also create a list of my own guidelines based on what I have researched on.

One practice that is among all three sources is branching strategies. The source from Atlassian, focuses on the Gitflow Workflow. Their focus on the workflow really emphasizes the use of branches such as feature or develop and also pull requests. In my second source, GIT-SCM, they also focus on feature branching but they also have different models that are offered. The difference is that no one model is recommended more than the other. GitLab also focuses on Gitflow Workflow like Atlassian but they also use hotfix branches to expedite any critical issues. Gitlab also has feature branches and release branches.

Another practice that all three sources agree on is how they commit their code and making sure it is purposeful. Atlassian and GIT-SCM think it is important to have small pushes

that are more focused. They regularly have commits of their code. Gitlab also promotes small atomic commits like the other two and also code reviews that have merge requests.

Reviews and Pull Requests are also another practice used in version control guidelines. Atlassian integrates pull requests into their standard workflow for peer review. Git-SCM acknowledges pull requests as a tool for collaboration. This is especially found in their open-source projects. Gitlab has a process called merge requests as previously mentioned. Their philosophy is that: **"Code review is essential for improving code quality, ensuring security, and building shared understanding within teams."** They have features such as inline comments, approval rules, and draft merge requests.

There are some more outdated guidelines that I have found. For example, strict branching models. This type of branching might be too unyielding as version control guidelines are always evolving. Another outdated guideline is rebasing. Rebasing allows for a clean history but it also has the disadvantage of safety and traceability.

For my version control guidelines, I would make sure to emphasize several things.

1. **Make Atomic Commits** – Helps with debugging and making sure to keep track of any changes.
2. **Write Descriptive Commit Messages** – For better collaboration.
3. **Use Branches Strategically specifically Feature Branches** – Keeps development clean.
4. **Integrate Changes Frequently** – Reduces conflict complexity.
5. **Review Code Together** – Improves quality and team knowledge.
6. **Avoid Committing Broken Code** – Keeps the repository stable.

By following strong version control practices, developers can maintain a clean and collaborative codebase. The core principles which are clean commits, thoughtful branches, and proactive merges remain essential for effective teamwork and making sure the projects are successful with the team.

Sources:

Atlassian Git Best Practices

https://www.atlassian.com/git/tutorials/comparing-workflows

Git-SCM Official Docs – Pro Git Book

https://git-scm.com/book/en/v2

Gitlab

https://about.gitlab.com/blog/2023/07/27/gitlab-flow-duo/