Truman Forey

CSD-380

6/15/2025

***Version Control Guidelines***

LOREM IPSUM

Whenever a software development team decides to begin working on a major new project, an important tool that needs to be used is version control. Version control tools, such as Git or SVN, allows developers to have greater control over the state of their project and acts as a layer of safety for making sure that the project doesn’t get deleted or irreversibly corrupted. However, these tools are a magic cure-all – properly using version control requires understanding and establishing best practices and understanding why they’re important. And while everyone’s idea of best practices will be different in some way, there are a number of consistent ideas that we can learn from and build unique policies on top of.

One of the most valuable things to standardize are code reviews. Reviews should be done regularly, both at a large scale to be sure the project is in a stable sate and before committing any changes to a shared repository. This helps to minimize bad code being put into the project early, which stops bad code from propagating and spreading later in the project’s lifespan, which is harder to remove. BY extension, it’s also a good idea to only make small, incremental changes at a time – this minimizes the damage a single bad commit or push can do. Finally, branches should be used in clear and consistent ways, such as by only using them for experimental features or releases. Branches are powerful tools, but they require group understanding and coordination in order to make sure everyone knows how to use them correctly.

However, the singular most consistent best practice comes when it’s time to commit changes. Since commits are the point where the main project becomes altered, it’s absolutely important that they’re handled delicately. One of the most important things to do is to make sure that committed changes are made ‘atomically’ - as a complete batch. If you’re preparing to commit files but some of them are incomplete or unable to be committed, then the entire commit batch needs to be abandoned until the issue is resolved. This is because if incomplete commits are performed, it could lead to a cascade of issues as the repository tries to work off of half-modified files. But there’s one incredibly simple action that everyone needs to be responsible for: writing clear, descriptive commit messages. One of the most important parts of using version control is keeping track of what’s been included in each commit and what it’s accomplishing. That way, any reversions that needs to be done can be quickly tracked down and executed. Writing vague or unhelpful commit messages can only cause unnecessary confusion and strife.

However, there are also a large amount of smaller tips that can are worth considering – not everyone will agree they’re necessary, but they have valid reasons to consider implementing into a version control plan. For one, it’s worth stating the obvious that a version control plan should be well-planned , implemented, and kept up-to-date as the needs of the project changes. While version control tools exist to help manage projects, they are also projects themselves and need to managed as well. This also includes maintaining any documentation related to the project. In addition, it’s also important to create a good system of security around a version control system, such as making sure that employees only have authority to modify the files that they are directly responsible for. Finally, it’s important to regularly test any recovery/reversion processes to make sure they’re effective. That way, they can be relied on to be effective during actual recovery scenarios.

Sources:

<https://about.gitlab.com/topics/version-control/version-control-best-practices/>

<https://www.perforce.com/blog/vcs/8-version-control-best-practices>

<https://daily.dev/blog/documentation-version-control-best-practices-2024>