Jason Schriner

Module 3

6/11/25

The software industry requires rapid development to meet customer demands and to stay competitive. Tools like version control can help to achieve this. Version control allows developers to make rapid changes and quickly share these changes with the rest of their teams. The ability to share is an important part of version control as it keeps teams from becoming siloed. Version control is an important tool in modern software development and as such there are some common best practices that are recommended to follow.

GitLab, Perforce, and Modern Requirements agree to the following best practices. First, you should only make small changes to the repository. If, for example, you had three objectives, you would want to complete one objective and then push that code before beginning the next objective. Keeping tasks separate helps to make code review easier. Next, you should create clear commit messages. If you were fixing a bug, you would want to put that in the commit message along with any relevant information pertaining to the bug. This allows anyone to understand the reason for the commit later. When working in a team, it is also important to create a strategy for creating and merging branches. There are multiple approaches that are commonly used. One strategy is to create a new branch for each feature in development. Another strategy is to create one for each developer. It is important that everyone understands and follows the same strategy. It is possible to customize any strategy if needed based on your company's needs. However, you should follow the KISS method and only make changes to one of these strategies if it provides some real benefit to the team.

Perforce and Modern Requirements both agree that you should also implement some type of access controls for security. A code repository can include valuable assets that need to be protected from theft and tampering. Modern Requirements add to this by recommending that you utilize backups. This is always a good idea, but this recommendation is a little more dependent on the type of version control that you are using. A centralized version control system has all of your code in one place, and in this type of version control it is a good idea to create backups. In a decentralized version control system, everyone has a copy on their own system creating multiple backups. They also recommend using version numbers. Version numbers can help you keep track of your code and ensure that you release the correct version of your code.

Perforce also recommends that you include all documentation and files necessary for any commit. This will ensure that others can fully understand and utilize the changes without missing any dependencies or context. They also recommend that you perform code reviews before committing anything to a shared repository. This will help maintain code quality and allow you to catch bugs or issues earlier in the development process. It also fosters a sense of responsibility among the team. Their last recommendation is a very strong one, it is do not break builds. What they mean by this is that you should only commit code that is complete so that it will run on another person's system. This may require that you provide test cases so that it will be usable by others without breaking their builds. The last recommendation is from GitLab, and it is to obtain feedback from code reviews. First, this creates a collaborative environment where developers can learn from each other. It also helps to improve code quality.

After reviewing the recommendations from GitLab, Perforce and Modern Requirements, I have identified a list that aligns with my personal approach for effective use of version control. The first one is to make small commits to ensure that it is easy to review code later. Next is to create clear detailed commit messages that make it easy to understand what you are doing and why. Additionally, you should ensure that you commit everything necessary such as documentation and test cases. It is also important to create a branching strategy for everyone to use to ensure that everyone is on the same page and there is no confusion when working with branches. The next recommendation is to have some type of security measures or access controls in place to protect your repository from theft or sabotage. I also like the idea of code reviews to not only improve quality but to help share knowledge and build collaboration. My last recommendation is to ensure that you utilize some form of backup to protect your codebase from any type of loss or corruption.

Version control is more than just a tool for managing code; it is a critical part of building software. As development teams grow and projects become more complex, following proven best practices becomes essential to maintaining code quality, avoiding confusion, and protecting valuable assets. By adopting some of these best practices, teams can improve not only the stability and security of their codebases but also the overall collaboration and productivity of their development process.

References:

Satpathy, A. (2023, August 21). Version Control Best Practices. Modern Requirements. <https://www.modernrequirements.com/blogs/version-control-best-practices/>

8 Version Control Best Practices | Perforce Software. (2024). Perforce Software. <https://www.perforce.com/blog/vcs/8-version-control-best-practices#version-control-best-practices>

What are Git version control best practices? (n.d.). GitLab. <https://about.gitlab.com/topics/version-control/version-control-best-practices/>