Zachariah King

August 31, 2025

Version Control Guidelines - Assignment 3.2

CSD 380

The first source in focus is from Medium and it's titled "Version Control Best Practices for Modern Software Development" and it was posted on January 2 of this year. The articles starts by stating how cracks in version control are bound to occasionally happen especially with inexperienced teams, but the author goes on to let us know the things they wish someone told them. The thorough piece throws in a refresher on exactly what makes up version control systems such as the repositories, the commits, and the branches. It then goes deeper by explaining the usual routes of branching, specifically for Git, and they are Git Flow and Github Flow. It also mentions the relevant trunk based development strategy as well where there's only one main branch that all developers commit their code to.

The article also mentioned how imperative adequate commit messages are. Simply put, trying to understand complex systems with weak, general updates is not going to be easy for anyone. The commit messages should contain clarity, specificity, reasoning, the imperative mood, conciseness, and a referenced issue ID number. Then pull requests are mentioned and it states that they should as well be short and sweet, explanatory, routes to specified issue areas, and addressed to the appropriate people who know about the project you're working on. Code reviews are also important and they should be useful, considerate, and highly detailed. They should also look for functionality and an easy-to-read nature that's still specific and positively practical. Automation is also brought up as an important factor in making these processes easier. More tips were mentioned such as rebasing and interactive staging, but they are for after one masters the general guidelines.

The next source I found is from Michael Ernst and it was last updated on April 8, 2024. It's titled "Version Control Concepts and Best Practices." This article also goes in detail on what

VCSs are and how they work and this one mentions the differences between distributed and centralized version control systems, although it did say that centralized systems are currently declining in popularity and/or relevance. This is probably because distributed systems originated in modern times which makes it operate faster and be less prone to errors while offering more features that come from an intricate behind-the-scenes system.

On the topic of specific advice on how to implement these systems, this source freely admits to the fact that these guidelines probably don't relate to super complicated situations where deeper analyzation is needed. This source does give plenty of great advice though, some of which that did match the first source such as using descriptive commits. Some unique information this article gave is to make sure teams are coordinating immensely to the point where everyone is incorporating their alterations frequently and to never *force* an operation, especially when the system is initially refusing to complete a task. This article gets deep and goes into specific tips for specific Version Control Systems which I thought might be helpful down the road so it was made sure to be bookmarked.

The third article I decided reference is titled "Version Control Best Practices for Efficient Software Development" by Arunabh Satpathy and it's from Modern Requirements. Being also fairly recently posted, I was able to be more trusting of this source's advice such as including some sort of version number in the title of the alteration one is making and possibly committing. It even goes into detail and includes the format it should be in which is MajorVersionNumber.MinorVersionNumber.PatchesNumber also known as Major.Minor.Patch Format. This article also goes into detail on how important security is and mentions access control mechanisms and tools for backups and failovers.

Overall, all sources emphasized exquisite documentation (commit messages, etc) and the countless benefits of proper Version Control System management as well as how some of the concepts are irrelevant nowadays like Centralized VCSs. They all had a little bit of information though that the other did not, so this shows just how information-heavy the concept of proper VCS use is and how we can always take some time to learn a little bit more about it. If I had to put together my own little list of best Version Control Guidelines, it would consist of these because I feel like they were the most emphasized in the sources I found and because I feel like the whole point of VCSs are to communicate with the team show skills independently while still working on the bigger project:

- Document everything!
- Remember you’re working with people so be kind!
- Remember you’re being watched (don’t be lazy).
- Be specific.
- Research your System with purpose to know how to take advantage!

References

Guy!!, S. E. (2025, January 2). *Version control best practices for modern software development*. Medium. https://medium.com/@paulosejithu/version-control-best-practices-for-modern-software-development-67bdfeddd74a

https://homes.cs.washington.edu/~mernst/advice/version-control.html

https://www.modernrequirements.com/blogs/version-control-best-practices/