Amanda Riley  
9/1/24  
CSD380-O316  
Module 3.2 Assignment

I used three sources for my research: “Documentation Version Control: Best Practices 2024” from daily.dev [1], “8 Version Control Best Practices” by Brent Schiestl at Perforce [2], and “Version control concepts and best practices” by Michael Ernst [3]. Interestingly, the first one was written this year, the second one was from 2020, and the third one, while updated recently, was first written in 2012. I thought this would help me answer the question of whether there might be outdated best practices and how much has changed in the past 12 years.

The daily.dev article is the one that I started with because I thought its currency would make it the most trustworthy. It emphasizes certain good habits and offers sensible explanations for doing so. For example, for security and availability reasons, you should always back up the data being stored, follow best security practices, and review documents for correctness. In order to actually make good use of version control in the first place (so that you are actually tracking your changes), you should always leave clear commit comments and should commit as soon as you reach a good stopping point and branch as soon as it makes sense (when it is basically a new version). And last but not least, there should always be a clear plan to implement all of this that everyone on the team is aware of.

The Perforce blog delineates eight very specific best practices, most of which were also mentioned in the first article at some point:

1. Always commit “atomically,” which means that each commit should represent a whole “piece” of the program (like an atom), no more and no less. In other words, it shouldn’t be incomplete and it definitely shouldn’t be multiple parts combined.
2. Each commit needs a single purpose that makes sense to the project, since the purpose is to make a *version* control that actually tells you something about the software’s development over time. It’s not simply a way for you to back up your work at the end of every day.
3. Always leave a useful comment with each commit – it should let the reader know *why* the change was made and even give the issue ID if available.
4. Do not break builds by doing an incomplete commit. Even if it passes unit tests, it might have integration issues.
5. Request code reviews before committing if the repository is shared.
6. Commits must be traceable. This is one of the basic definitions of version control.
7. Branching best practices: keep it simple, have a plan and owner, branch off for new releases.
8. Use security best practices to protect business assets and intellectual property.

The article by Michael Ernst distinguishes between version control in general and distributed version control, and offers some background information as opposed to just best practices. These best practices are also similar to the first two. He emphasizes the importance of good commit messages and atomicity. He also emphasizes good collaboration, making sure that you are paying attention to others’ commits and making them aware of yours. Finally, he gives some very specific tips that are helpful for beginners like me, for example making sure you understand the specific features of Git like the merge tool, if you are going to use it.

I would say there were more similarities than differences among these three articles, and because of that, I would say these general best practices haven’t really changed (and shouldn’t change) over time. My list of best practices would be very similar to a combination of all three of these, especially the first and last. I found them to be very comprehensive. Mostly I would emphasize making a good plan that everyone is aware of and on board with, and just *thinking* about what you’re doing at all times: is this ready for a commit? What is the ideal message that will be useful later on? This will ensure that you don’t fall into bad habits and ultimately get good use of your version control tool.

[1] daily.dev, “Documentation Version Control: Best Practices 2024,” Jul 2024. Available: <https://daily.dev/blog/documentation-version-control-best-practices-2024>

[2] B. Schiestl, “8 Version Control Best Practices.” Perforce, May 2020. Available: <https://www.perforce.com/blog/vcs/8-version-control-best-practices>

[3] M. Ernst, “Version control concepts and best practices,” Sept 2012. Available: <https://homes.cs.washington.edu/~mernst/advice/version-control.html>