Eric Williams-Phillips

Bellevue University | CSD 380

Sue Sampson

**Version Control Guidelines**

Abiding by a proper set of version control procedures can minimize the risk of error and eliminate clutter if done correctly. There are several sources out there that provide a great set of guidelines for ideal an ideal version control practice but the sources that I settled on were Washington’s School for Computer Science, Modern Requirements, and Perforce. Each provides a clear and concise list of good version control practices and includes an explanation of each along with their reasoning. Of the three sources, a practice that can be agreed on is the use of descriptive commit messages. Similarly to comments in the codebase, it’s important to be as transparent as possible when pushing changes as other reviewers and examiners cannot read your mind. Providing a detailed commit message can set the next person up for success and increase morale by requiring less “guess work”. Alongside a detailed commit message, they should be singular and logical units. Each commit should have a unique purpose. Rather than trying to “kill two birds with one stone”, WSCC suggests separating commits by purpose (bug fixes should be pushed separate from typo fixes for example). Another shared suggestion is to review the commits before making them to avoid indiscrimination. This can improve the overall quality of the repository by reducing randomness and unnecessary pushes. Automatically generated files such as .io or .class files should also be excluded from pushes as they are not usually necessary. Those types of files can easily be re-generated by the reviewer during a test. Avoiding including these can promote a cleaner and more focused repository. Another practice that Perforce recommends is to make sure each commit is traceable. Like the idea of leaving behind breadcrumbs in a forest, to avoid getting “lost” or dealing with a bug that is difficult to locate, it is a good idea to make sure the author and any reviewer comments are included. This practice can preserve the history of the repository and abides by common security principles as well. There are several more tips to promote good version control (version numbers, team coordination, merging strategies, etc) but my list of the most crucial practices would include: detailed commit messages, traceability, change discrimination, and inclusion of proper version number conventions. These seem to be the type of practices that are easily implemented and alone can save lots of time if done correctly. Including detailed commit messages is so important it almost seems like it should be a requirement. Ensuring each commit is traceable can save time and money and is also beneficial for security auditing purposes. Reducing randomness in commitments and avoiding committing generated files can keep a repository clean and organized to support a smooth development process and version numbering is a cherry on top when it comes to organization. I do believe all these practices are very important, however, and no guideline is perfect. Following suggestions from multiple sources can provide tips that may have been overlooked and ultimately get as close to perfection as possible.

Works Cited

Satpathy, A. (2024, July 11). *Version control best practices*. Modern Requirements. https://www.modernrequirements.com/blogs/version-control-best-practices/

Schiestl, B. (2020, May 21). *8 Version control best practices*. Perforce Software. https://www.perforce.com/blog/vcs/8-version-control-best-practices

*Version control concepts and best practices*. (n.d.). https://homes.cs.washington.edu/~mernst/advice/version-control.html