Jeremiah Kellam

CSD380 Module 3.2 Assignment

7/15/2024

Version Control Guidelines

Version control is one of the main items that is critical to today’s software development. It is something that helps software development teams to handle the way they manage changes to their code over time. Version control is a system or process that tracks and manages changes made to a specific item over time. These items can be documents, software, and other items and version control provides a trail for all changes to track and allows reversion to prior states if it is required (Korta). For the length of this assignment, the focus will be on version control guidelines regarding software development.

An interesting source that was found was by Eric Sink and written in 2008 at the start of developing source control items and defining best practices. It is a very good comparison of what was and what is. The tools today for version control weren’t available in 2008 and Sink writes about best practices where everything is a manual operation, and each person is the responsible party for their practices without a tool to aid them. The very first best practice he lists is “Don’t break the tree,” meaning that any code that is added must not break the entire code and allow others to continue their work or else a social penalty will be enacted to embarrass the guilty party to learn from their mistakes. The list of best practices that were applicable in 2008 that are not quite relevant in today’s industry due to Git or Subversion is to use checkouts/locks carefully but is still applicable when dealing with a centralized server with a single repository. Some items that match other lists that will be expanded on later are to describe and fully explain checkins/commits and their purpose, perform atomic transactions that checkin/commit single item changes so that integrity is maintained when compiled and are easily reverted if needed. Always stay up to date with current versions so that work completed is applicable and able to be integrated and only merge-on-get so that the party can compile and prove integrity before pushing to commit-to-main. Only store what is necessary for a build in the repository and nothing else. Checkin/commit often without breaking the build so your work is not lagging, and others don’t try to work on something already completed (Sinks).

Two sets of guidelines are short and cover roughly the same items. GitLab’s best practices include small incremental changes, atomic commits, use branches so that the main codeline is not affected, write descriptive commit messages, get feedback through code reviews using peers or AI, and specify the scheme of branching the project will follow whether it is centralized or by function or person (GitLab). The second short set of guidelines is from git-tower.com written by Tobias Gunther also suggests small commits that target a single item, to commit often but only complete work, test before committing, be very descriptive with commit messages, don’t use repository as a back-up system, utilize branches after deciding the workflow, don’t just use basic Git commands, and find a tool to support the use of your version control system (Gunther).

The last list of guidelines is from acompiler.com and is the most comprehensive set of 47 items written by Rajeev Bera. It is highly recommended to look at this work ([47 Git Best Practices to follow (in 2023) - aCompiler](https://acompiler.com/git-best-practices/)) because it not only lists so many items, it explains why it is good to implement. The top items include atomic commits, commit often, do not commit generated files / dependencies / config files / broken code, test before committing, write descriptive commit messages, review your commit, use reference numbers / hashes to track commits, don’t use repository as a back-up, decide on a workflow and enforce standards, use branches, and do not use refactoring is with a new feature (Bera).

One interesting device that was included at the end of the Bera’s guidelines is the AFTER technique.

A - Atomic Commits.

F - Frequent Commits.

T - Test before you push your changes.

E - Enforce standards.

R - Refactoring is not a new feature.

This technique in and of itself sums up nearly all the key suggestions of the different guidelines, regardless of when it was written. The contrasting parts of them were interesting to list as well. Sink’s guidelines were outdated but still had nuggets of gold when using the best personal practices. No other guidelines talked about working folders, but all others talked about using branches. Sink’s guideline was so early that the branching nomenclature was not in place yet. AI for code review simply did not exist, which was suggested in GitLab. Oddly enough all articles suggested using tools to assist version control and so happened to promote their own. GitLab was the most succinct but descriptive, Git-Tower was short but not very informative, Bera’s had the most does and don’ts with valid reasons, while Sink was the best for me to understand version control overall because it explained each part of it and how they interact before building up to today’s new possibilities.

My list for guidelines includes atomic commits, frequent commits, write descriptive commits, test before pushing your changes, don’t commit unnecessary generated files, set a standard and define a workflow, utilize branches, and perform code reviews after committing. Hopefully these guidelines are what will assist in being successful in personal projects and future endeavors.

References:

Korta, S. (2024, August 9*). 5 Document Versioning Best Practices & Guidelines [2024].* <https://www.imageapi.com/blog/version-control>

Bera, R. (2024, February 11). 47 Git Best Practices to follow (in 2023) - aCompiler. aCompiler. <https://acompiler.com/git-best-practices/>

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

Günther, T. (2020b, July 14). Version control best practices. Tower Blog. <https://www.git-tower.com/blog/version-control-best-practices/>

Sink, E. (2008, August 27). *Source Control HOWTO.* <https://ericsink.com/scm/source_control.html>