Version Control Guideline

Robert Dylan Bean

Bellevue University

CSD:380 DevOps

Professor John Woods

Assignment 3.2

April 6, 2025

**Introduction**

Version control has become a key tool for managing changes and supporting smooth collaboration in both creative and technical work. Whether one is editing a report, technical document, or code, version control helps keep track of updates, prevents accidental mistakes, and makes it easier for teams to work together. I’ve chosen three sources to compare the different technics and elements of how they each go about addressing version control. Filestage, the Daily.dev guide, and Nulab’s ultimate guide are all excellent resources and are compared and contracted with each other below.

**Filestage: “Document version control best practices and examples”**

The Filestage article focuses on managing documents when multiple people are involved. It recommends using clear and consistent naming formats like “Document\_Name\_v1, v2, v3,” so everyone knows which version is current. It also talks about setting up organized review rounds, collecting all feedback in one place, and comparing document versions side by side. These steps are meant to make teamwork less confusing and avoid wasting time searching for the latest version or tracking down edits.

**Daily.dev: “Documentation Version Control: Best Practices 2024”**

The Daily.dev guide takes a more technical angle, especially useful for software and technical writing. It talks about having a clear version control plan that includes writing helpful commit messages, using branches to work on new features, and setting up automation to keep documentation updated. It also mentions keeping documentation in sync with code, having regular backups, and using security measures. Even though it's focused on software, the ideas still apply to any kind of team writing project that needs to be clear, secure, and up to date.

**Nulab: “The ultimate guide to document version control”**

Nulab’s article explains version control in a straightforward and easy-to-follow way. It compares it to a "time machine" that protects your work and keeps teams on the same page. The guide walks through steps like starting repositories, making commits, creating branches, and solving conflicts without getting too technical. It also gives advice on how to choose between centralized or distributed systems depending on your team’s needs, and it recommends setting up workflows and training to help everyone use the system well.

**The Comparison**

When comparing all three sources, there are a few shared ideas that stand out. Each one stresses using clear naming rules and keeping good records of changes. They all agree that strong teamwork is important, whether that means using tools that allow branching and merging or just collecting feedback in a central spot. Some older habits, like avoiding labels like “final,” are becoming less important because modern version control systems automatically handle those details. That means teams can put more energy into their work and less into managing versions.

**Personal Guidelines**

1. **Consistent Naming Conventions**: Use a simple and clear naming format so it's easy to tell which version is the latest.
2. **Clear Workflow and Review Cycles**: Set deadlines and feedback rules to avoid everyone editing at the same time.
3. **Centralized Feedback Collection**: Gather all notes and comments in one place to make it easier to follow.
4. **Regular Commits and Clear Messages**: Save updates often and use helpful commit messages so others know what was changed.
5. **Branching and Merging Practices**: Create branches for trying new things and only merge once they’ve been reviewed carefully.

**Conclusion**

These are the guidelines I’ve chosen because they tackle common teamwork problems, like confusion over versions or duplicate edits. They help keep everything organized, promote accountability, and make sure everyone’s working together effectively. Whether anyone is writing stories, building software, or editing reports, having a good version control setup seems to make a big difference.

**References**

Daily.dev. (2024, July 15). *Documentation version control: Best practices for 2024*. <https://daily.dev/blog/documentation-version-control-best-practices-2024>

Filestage. (2025, February 13). *Document version control best practices and examples*

<https://filestage.io/blog/document-version-control/>

Nulab. (2023, November 15) *The ultimate guide to document version control*. <https://nulab.com/learn/collaboration/document-version-control/>