Jaci Brown

380 Mod 3.2

Version Control

[NIH Version Control Guidelines](https://files.nccih.nih.gov/s3fs-public/CR-Toolbox/Version_Control_Guidelines_ver2_07-17-2015.pdf)

The NIH page seems to be a file for their preferences on internal documents. They provide detailed descriptions with examples of naming conventions, as well as a flow chart to summarize and outline further. Their top points can be summarized as follows:

1. Keep dates of creation and revisions on front page and in header/footer of all pages.
2. Keep Version numbers up to date on the first page and header/footer of all pages if possible.
3. Keep the same process for draft versions and final versions.
4. Use numbers after the decimal to denote revisions and go to the next whole number after a revision is approved/complete. Ex. V1.3 = under revision for the third time. V2.0 = next fully approved version.

[Nulab || The Ultimate Guide to Document Version Control](https://nulab.com/learn/collaboration/document-version-control/)

Nulab is a service provider for organizing backlogs, team collaboration, and drafting detailed presentations. Their article on version control is more directed towards software rather than general business documents like NIH. The article walks through version control as if the reader had no experience, starting with why it is important and how to establish it. Their guidelines include:

1. Knowing the needs/goals of the business/users of the system.
2. Pick using centralized or distributed system based on need. They recommend distributed systems, like Git, for larger projects where needs may vary more for access.
3. Set up your chosen system and test.
4. Establish a workflow and document requirements such as frequency of uploading changes, file naming preferences, and group editing protocols.
5. Seek feedback to identify problems or potential methods for improvement.

After the system is in place, they recommend establishing more concrete rules like starting at version 1, incrementing by decimal for small changes, including dates of changes, and a description of the changes made. Nulab also highlights the importance of consistency, meaningful update notes/commits, and to uses branches sparingly to reduce ‘clutter’ in the repository.

[Perforce || 8 Version Control Best Practices](https://www.perforce.com/blog/vcs/8-version-control-best-practices)

Perforce is the provider of Helix Core, a version control system. They highlight the practices they cover as the most critical.

1. Commit fully (all documents changed) or not at all to prevent unfinished documents from being “live”.
2. Commit for a reason, like a new feature or bug fix. Not just to “update” as it makes rollbacks more difficult to pinpoint.
3. Have clear messages stating why there were changes. Also include any ticket numbers if related to the commit.
4. Keep all updates traceable. Include names of authors, editors, and code review comments.

While the NIH article was reflected more toward ordinary documents and not code, it still followed the naming conventions mentioned in the two latter articles. All sources prioritized accountability in the way that all changes and who made them should be documented and available. The two software specific sources recommend clear commit messages to help identify problems that could require rollbacks. In addition to agreeing on commits and traceability, both software sources approach branching with a ‘only as necessary’ view. The context they apply around the guidelines implies that branching can cause many problems if not done carefully and linearly.

Approaching the topic of relevancy, all ideas appear relevant to me. Consistent naming and providing clear messages are essential for team communication and ease of workflow. Additionally, committing fully and only completed documents/sources can help prevent breaking any live code.

In my opinion, the most important rules would be:

1. Have consistent file naming for neater repositories.
2. Having clear commit messages with all applicable case/ticket numbers for easier reference and rollbacks.
3. Document all authors, especially for group writings with a single commit for ease of communication if problems arise.
4. Stick to the established rules to keep everyone on the same page.
5. Provide feedback and allow for changes that improve the functionality of the team.