Version Control Guidelines

Version control systems are essential for managing codebases in software development, ensuring collaboration, version tracking, and error recovery. We can identify best practices by researching guidelines from various sources and evaluating their relevance in modern development.

Comparison of Guidelines

1. **GitHub Guidelines**
   GitHub emphasizes clear commit messages, frequent commits, and using branches for features and bug fixes. Developers are encouraged to follow a branching strategy, such as Git Flow, to maintain a clean main branch.

2. **Atlassian Guidelines**
   Atlassian recommends defining a branching model, such as trunk-based development or feature branching. They also stress the importance of code reviews before merging changes and keeping branches small to minimize conflicts.

3. **Microsoft DevOps Guidelines**
   Microsoft suggests tagging significant releases, setting up automated testing and continuous integration pipelines, and maintaining a strict access control policy for repositories. They also emphasize using descriptive branch names.

Relevant and Outdated Guidelines

Most guidelines remain relevant today, but some older practices, like manually merging changes or avoiding automated testing, are no longer efficient. With advancements in CI/CD pipelines, automation has become a cornerstone of modern version control practices.

My Version Control Guidelines

Based on the research, I propose the following guidelines:

1. **Frequent Commits**
   Commit changes frequently to ensure a detailed project history and reduce the risk of losing progress.

2. **Clear Commit Messages**
   Use descriptive messages to document what changes were made and why.

3. **Branching Strategy**
   Adopt a branching model that suits the project, such as Git Flow or trunk-based

development, to organize work efficiently.

4. **Code Reviews**
   Peer reviews are required before merging changes to maintain code quality and prevent errors.

5. **Automated Testing**
   Integrate automated testing to catch bugs early and ensure new changes don't break existing functionality.

6. **Access Control**
   Restrict access to critical branches to authorized users only, safeguarding the main codebase.

These guidelines foster collaboration, maintain code integrity, and streamline workflows. They were chosen for their relevance in modern software development, where automation, collaboration, and security are paramount.

Sources:

Mijacobs. (n.d.). Devops Resource Center - Azure Devops. Azure DevOps | Microsoft Learn. https://learn.microsoft.com/en-us/devops/

Atlassian. (n.d.). Atlassian Bitbucket: Code & CI/CD on the atlassian platform. https://www.atlassian.com/software/bitbucket

GitHub.com help documentation. GitHub Docs. (n.d.). https://docs.github.com/en