**VERSION CONTROL SYSTEMS**

* **Git:** Developed by Linus Torvalds in 2005. Git uses a distributed model, meaning each developer has a complete copy of the repository, enabling offline work and easy branching and merging. Known for its speed, efficiency, and strong community support.
* **Subversion (SVN):** Created by CollabNet in 2000. SVN uses a centralized model, meaning meaning that developers don’t download the whole repo and instead every command depends on one being connected to the internet. It’s known for its simplicity, compatibility with older systems like CVS, and strong community support, but it’s also known for lacking offline support and not being very fast. The creator of Git based a lot of the founding decisions in opposition to CVS (which is what SVN is based on), stating it as an example of what not to do.
* **Mercurial:** Developed by Matt Mackall in 2005 and is used by Mozilla. Mercurial uses a distributed model like Git. It’s known for emphasizing simplicity and ease of use while offering branching, merging, and distributed workflows.
* **Perforce:** Commercial version system designed for large scale development products. Perforce uses a centralized model like SVN. Perforce does commits “atomically”; when you submit changes to a group of files, either all the changed files are merged simultaneously, or none of them are. If conflicts result from multiple users working on the same files, these conflicts must be resolved before Perforce will accept the changes. An advantage of using Perforce is its ability to handle large-scale software development projects and provide high-performance version control capabilities, but a drawback is its commercial nature, which may entail higher costs and licensing fees for organizations.
* **Team Foundation Server (TFS):** Microsoft product that combines version control, project management, and build automation. It offers both centralized and distributed version control options. TFS provides a wide range of features, including work item tracking, code reviews, continuous integration, and release management. It integrates well with other Microsoft development tools and is often used by teams working with Microsoft technologies. This is also a con though, as it does not work well with more diverse environments and non-Microsoft products.