1. The main difference between the two classes is that Centralized VCSs keep the history of changes on a central server from which everyone requests the latest version of the work and pushes the latest changes to. This means that everyone sharing the server also shares everyone’s work. Sourceforge.net uses this type of versioning in their projects.

On the other hand, on a Distributed VCS, everyone has a local copy of the entire work’s history. This means that it is not necessary to be online to change revisions or add changes to the work. “Distributed” comes from the fact that there isn’t a central entity in charge of the work’s history, so that anyone can sync with any other team member. This helps avoid failure due to a crash of the central versioning server. Open source projects, such as Mozilla Firefox, tend to use this type of versioning.

After having worked with both of these systems (SVN and Git) my personal impression is that there is not such thing as a clearly superior system. On one hand, a Centralized VCS allows more access control via folder permissions or by allowing checkout of just the needed sub tree from the repository tree. Also these systems are more prone to be backed up regularly, allowing for a more reliable environment of work. The quantity of development tools and Integrated Development Environments (IDEs) integrating with them is considerable. On the other side, a Distributed VCS, like Git, is considerably faster and easier to work with when it comes to creating and merging branches and tags. Also, they allow for flexible workflows, tailored to one’s project and team needs.

However, both systems have room for improvement. Tools like SVN are slow when it comes to process transactions and working with tags and branches can be very tricky as well. Also, when someone on the team checks-in buggy code that takes down the project’s application, the whole team’s work is potentially affected since everyone has to have the same codebase.

However, a Distributed VCS, on the other hand, doesn’t allow for partial check out of the repository. Also, integration with many IDEs and other development tools (like file comparers) are not yet the rule nowadays and, since there are little repository managers with a Graphical User Interface for Distributed VCSs, it can be very hard in the beginning to be efficient and proficient with this type of Distributed VCSs.

There are some other factors to consider before choosing a VCS like the workflow you’ll be using, available hardware, etc. However, the current trend is to switch to Distributed VCSs to manage coding projects since VCSs like SVN are growing obsolete quickly. Try Git on your next project (not only coding project, it can be any type of collaborative project like a book or a paper) and you will realize the great advantages this system provides.

2.Distributed VCS: Git, Mercurial.

Centralized VCS: CVS, Perforce, SVN.

### 3.Maintaining different versions

### **Branching**

### **Access from any computer**

4.Staged,Unstaged,Comitted.