## SVN control

### Repository structure

Project under version control, beside source code folder, should also have *doc* directory for versioning any documentation and binding that documentation to the given source release. Additionally there should be a file called *changes* or *release-notes* which should give information about release version, date, implemented features and fixed bugs. Bugs and features should have appropriate numbers from issue tracker.
Any other files and folder which are related to the project and can be changed should also be versioned.

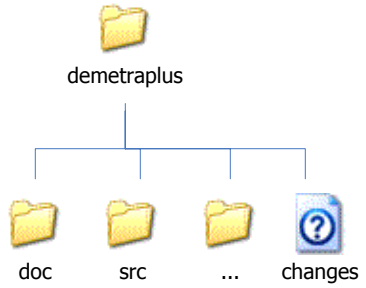


*Figure 1 Project structure in the SVN*

### Branching, merging, tagging

There are many branching strategies, which usage depends on many project factors like number of development teams, number of features, complexity of features, existence of customizations for different customers, etc. For SA Tool Kit project the most common, simple strategy with main development in trunk and branch per release is suitable enough.

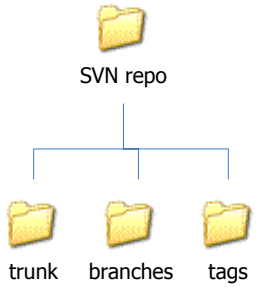


*Figure 2 SVN tree*

All features should be developed in trunk. This allows better tracking of changes and testing, since the code from trunk is supposed to be continuously built and tested in the CI environment.
However if there is justified need, temporary branch can be created, for example when there is a complex feature requiring multiple unstable commits causing broken builds. Code in trunk should always build successfully. When code is stable enough it should be merged into the trunk and branch should be deleted.

Release branches should be created for each major (e.g. 1.0.0) or minor (e.g. 1.1.0) release. For micro release (e.g. 1.1.1) minor branch should be used (e.g. 1.1.0). Branches are created in the *branches* folder. Branch should have name of the release. After branching, *changes* files should be updated.
Basically there should be no development in the release branches. Bug fixes should be applied to the trunk and then merged to the branch release. However there might be a situation in which it is better to fix a bug in the release branch and then merge it to the trunk. It's up to the developer to decide.

Tagging should be applied always when new release package is created. Tags are created in the *tags* folder of SVN repository.

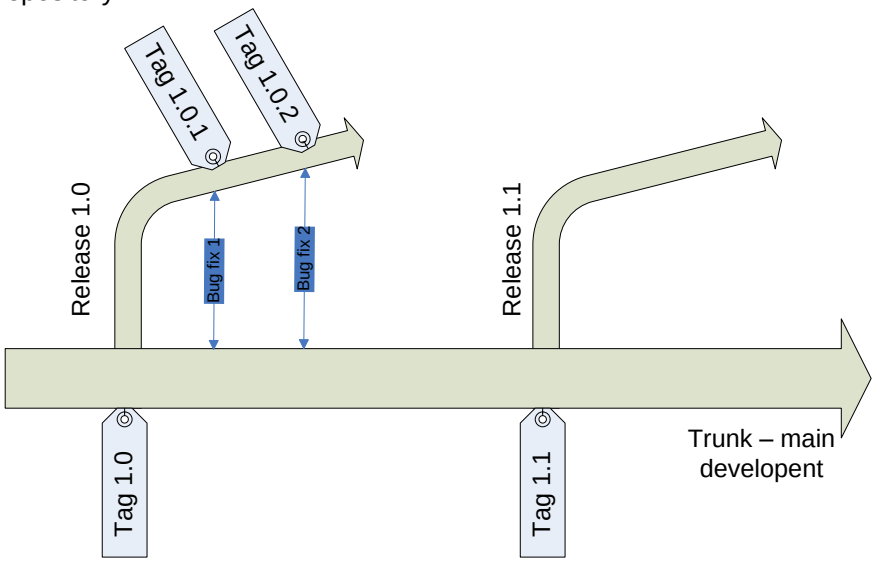


*Figure 3 Branching, merging and tagging strategy*

## Branching and merging Anti-Patterns

Proposed branching strategy can be modified depending on the current needs. However any approach should be though over to avoid common problems. Here is a list of anti-patterns which should be avoided when working with branching:

- Merge Paranoia – avoiding merging at all cost, usually because of a fear of the consequences.
- Merge Mania – spending too much time merging software assets instead of developing them.
- Big Bang Merge – deferring branch merging to the end of the development effort and attempting to merge all branches simultaneously.
- Never-Ending Merge – continuous merging activity because there is always more to merge.
- Wrong-Way Merge – merging a software asset version with an earlier version.
- Branch Mania – creating many branches for no apparent reason.
- Cascading Branches – branching but never merging back to the main line.
- Mysterious Branches – branching for no apparent reason.
- Temporary Branches – branching for changing reasons, so the branch becomes a permanent temporary workspace.
- Volatile Branches – branching with unstable software assets shared by other branches or merged into another branch.
- Note Branches are volatile most of the time while they exist as independent branches. That is the point of having them. The difference is that you should not share or merge branches while they are in an unstable state.
- Development Freeze – stopping all development activities while branching, merging, and building new base lines.
- Berlin Wall – using branches to divide the development team members, instead of dividing the work they are performing.