# Version Control, Part 2

**Diomidis Spinellis**

**maintenance branch:** A branch where most development concerns bug fixes.

**merge:** To combine different changes to the same file. Many systems follow the optimistic strategy of combining all lines that don't conflict.

**merge from current:** To merge changes from the current branch into the stable branch(es). To avoid disruptive changes in a stable branch, code changes are typically first introduced into the current (development) branch, tested, and then merged back.

**module:** A set of source code files under version control that can be manipulated together as one.

**pointy hat:** A virtual headgear element, similar to a dunce cap, humorously awarded to committers for acts of carelessness or stupidity.

**problem report:** see *issue* (Sept./Oct. 2005).

**release:** A software version that's made formally available to a wider community. Typically, each release is identified with a unique tag and often starts a separate maintenance branch.

**release engineer:** The person responsible for coordinating development toward a release. The release engineer will monitor pending issues for a given release, oversee the code freeze, and tag the release once it gets out the door.

**reparent:** To move the changes developed under one branch into another. The changes aren't committed to the original branch.

**repo bloat:** Changes recorded in the repository that don't contribute anything useful to the project's history (for example, a large erroneous commit and its associated back-out operation). A repo master can reduce repo bloat through repo surgery.

**repo master:** The person in charge of the version control system's repository; one who has rights to perform repo surgery.

**repo surgery:** Changes made directly to the version control system's repository, bypassing the system's commands—for example, by issuing database commands or directly manipulating system files. Through repo surgery, a repo master can perform operations that the version control system doesn't directly support.

**security branch:** A branch, created at the time of a release, to which only security commits are made.

**slush** (referring to code or features): A preparation for a feature or code freeze. During this period, developers will commit code they've been working on but are discouraged from starting on new elements. If a freeze lasts for a long time, a slush might be introduced to ease its passing by allowing in some extra elements.

**stable branch:** A branch where stability-disrupting changes are discouraged; the branch used for releasing the product's stable production version.

**synchronize:** 1) To pull the changes made in a parent branch into its (evolving) child (for example, feature) branch. 2) To update a view with the current version of the files in its corresponding branch.

**tag:** A symbolic name assigned to a specific release or a branch; provides developers and end users with a unique reference to the code base they're working with.

**tag slide:** To apply the same tag to a changed version of a file to correct a last-minute error found in a release.

**tinderbox:** An automated build and regression-testing tool. A tinderbox will typically fetch on a regular basis the latest versions of the software from each supported branch, build it for the different platforms, and report the results from the build and the regression tests.

**trunk:** The software's main line of development; the main starting point of most branches. You can often distinguish the trunk from other branches by the version numbers used for identifying its files, which are shorter than those of all other branches.

**vendor branch:** A branch for keeping track of versions of imported software. Differences between successive versions can then be readily applied to the locally modified import.

**version:** An identifiable instance of a specific file or release of a complete system.

**view:** A developer's copy of a branch. Changes made to the view are eventually committed to its corresponding branch. A snapshot view will hold the sources as they were at the time when they were retrieved and requires an update to synchronize them with the current version; a dynamic view always keeps the files synchronized with the current version.

For a complete version control glossary, see www.spinellis.gr/tot/vcs-glossary.html. ⓢ