

Version Control

CS 490MT/5555, Fall 2015, Yongjie Zheng

Fundamental Concepts

- ▶ **Version Control**

- ▶ The management of evolution of software systems.

- ▶ **Repository (Depot)**

- ▶ The place where all completed work is stored, including
 - ▶ The current state of the project;
 - ▶ The entire **history** (i.e. changes) of the project: when each change was made, who made it, and a text log message.

- ▶ **Working copy**

- ▶ The local reflection of the repository's files and directories.

- ▶ **Baseline / Trunk**

- ▶ The “primary” location for code in the repository. Think of code as a family tree — the “trunk” is the main line.

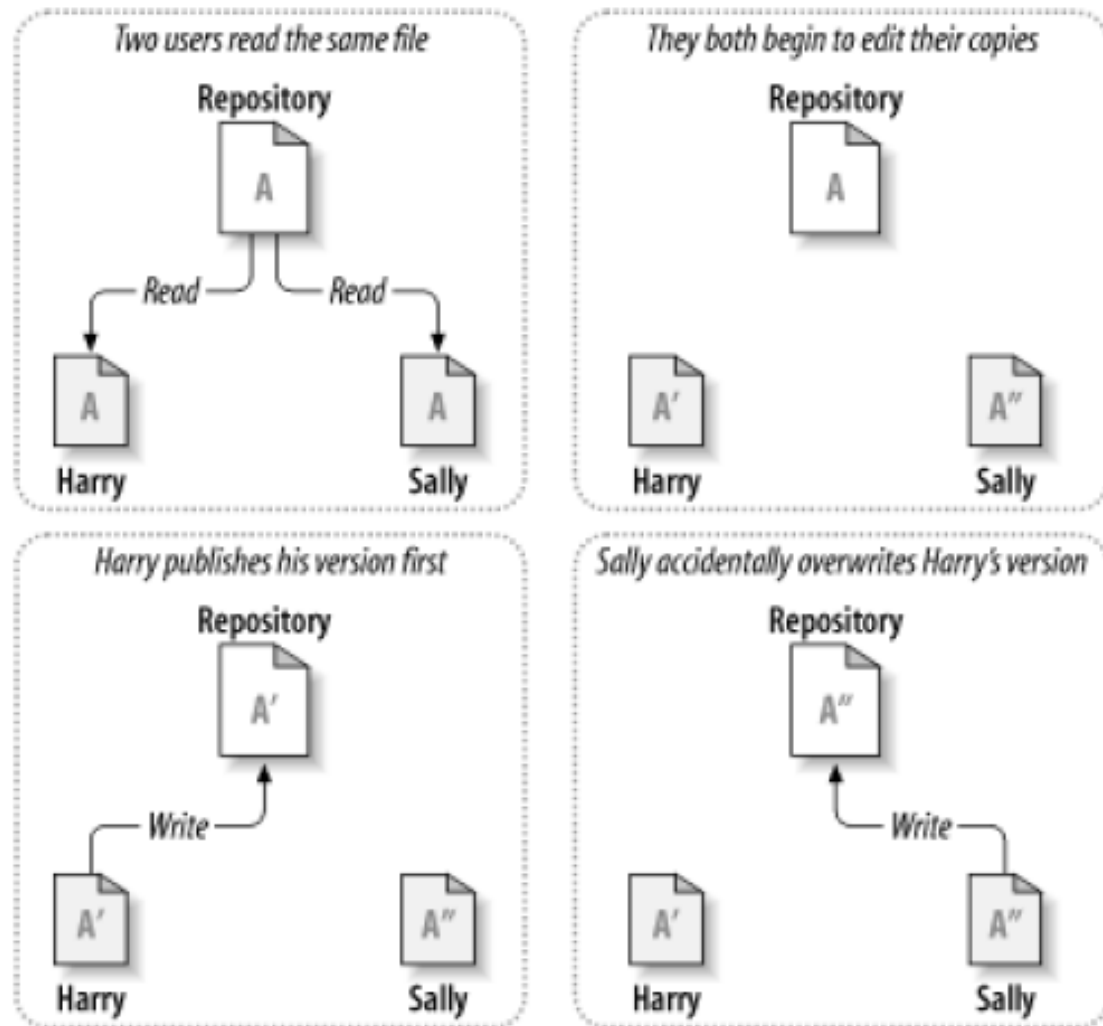
Version Control

▶ Mission

- ▶ Keep track of all old versions of files
 - ▶ Every file in the system has a full history of changes, and can easily be restored to any version in its history. Each version has a unique identifier that looks like a string of letters and numbers.
- ▶ Enable collaborative editing and sharing of data
 - ▶ How to prevent users from accidentally overwriting each other's changes?
 - ▶ Help developers communicate their changes to each other.

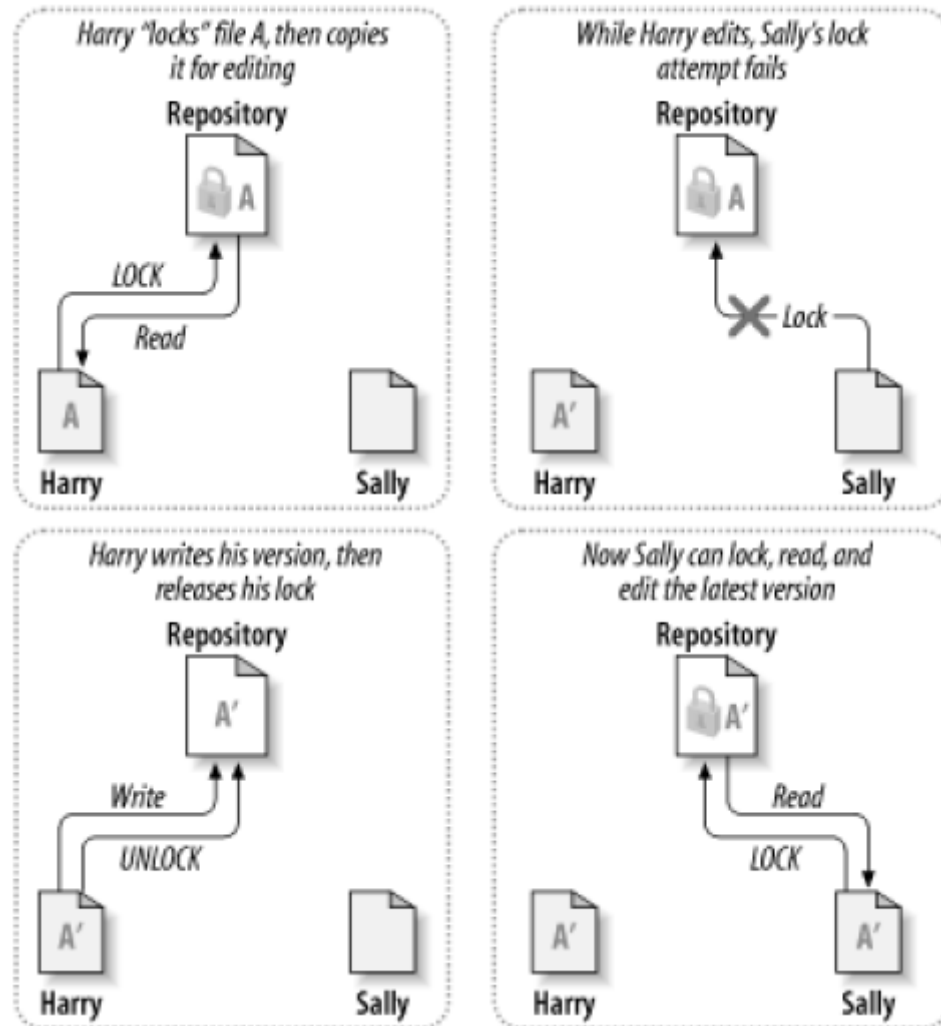
The problem of sharing

As a result,
Harry's work
is lost in the
latest version
of the file!



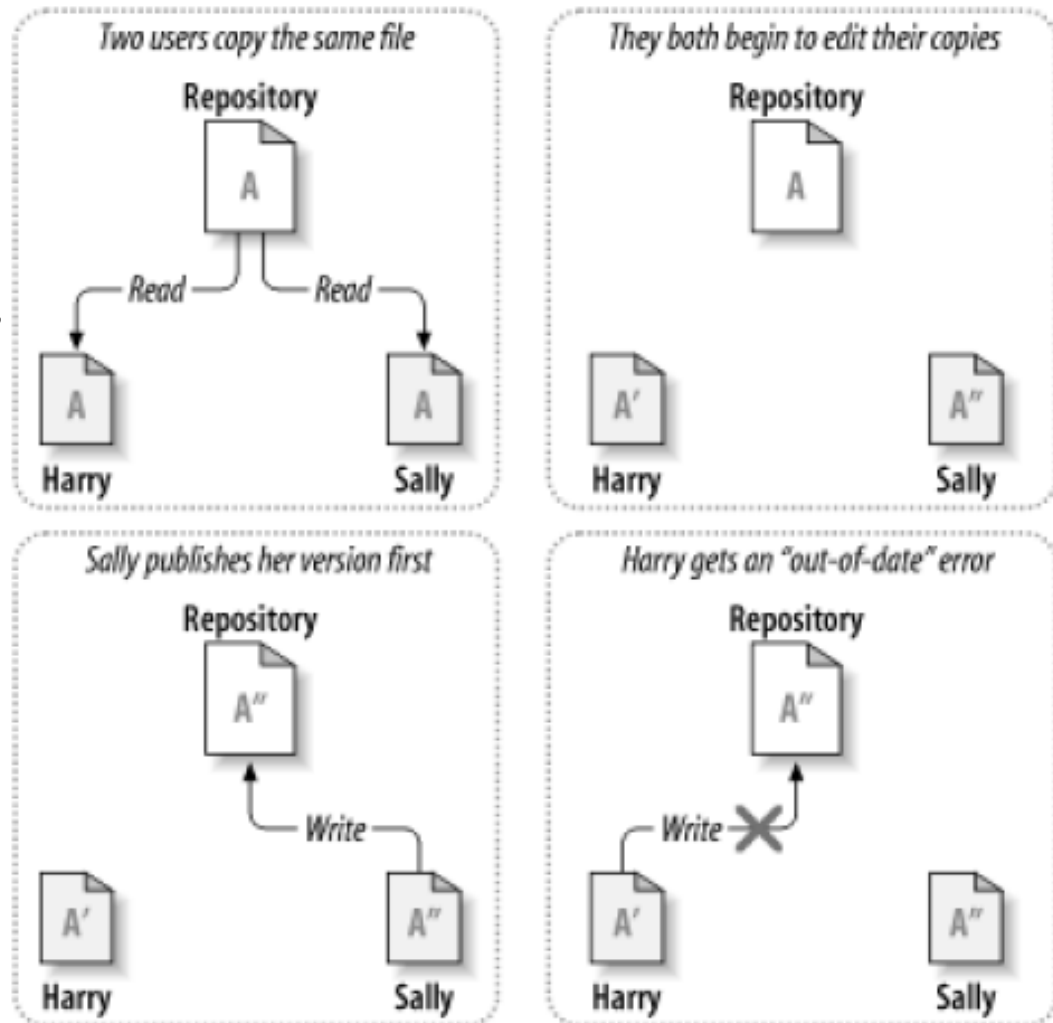
The lock-modify-unlock solution

- ▶ The repository allows only one person to change a file at a time.
- ▶ One must “lock” a file before making changes to it.
- ▶ The user can only read the file that is locked by others.



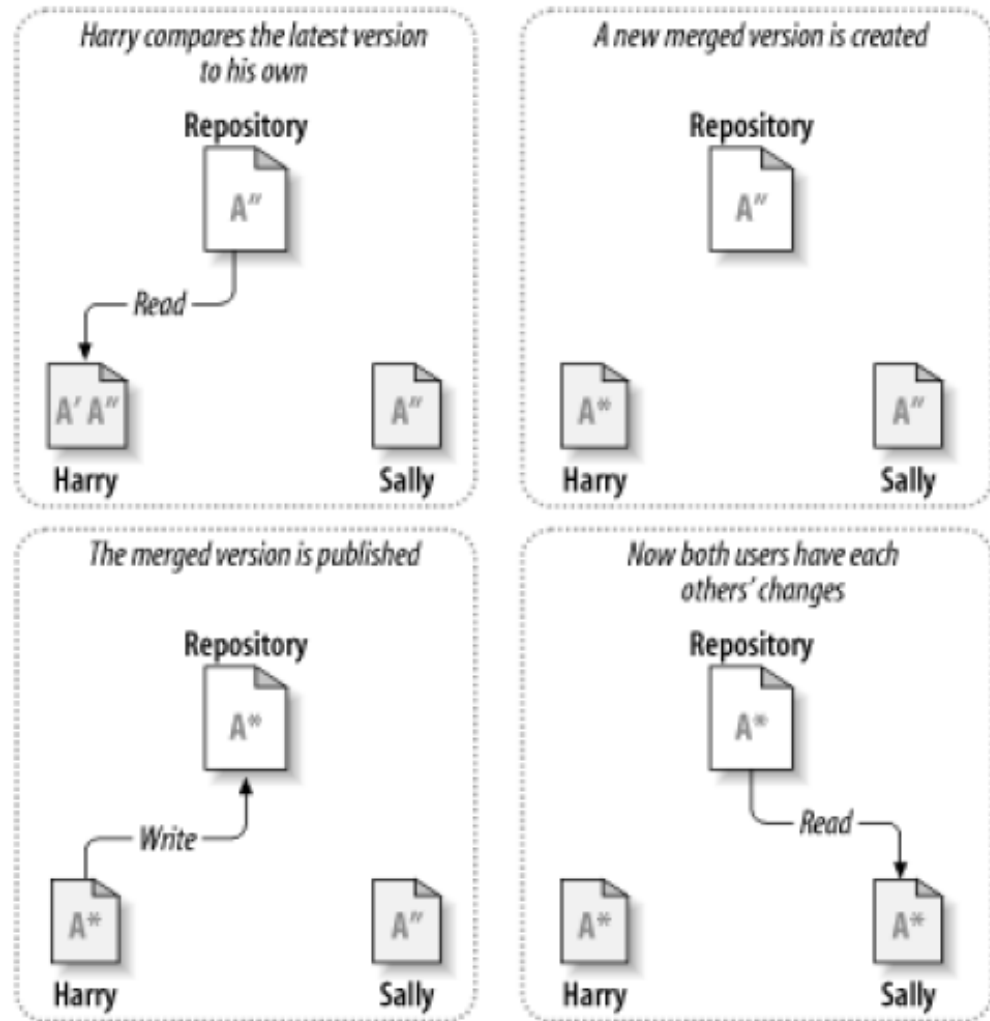
The copy-modify-merge solution

- ▶ Users can work simultaneously and independently on their private copies of the same file.
- ▶ The repository will inform the user that their file is out of date when the user is trying to upload their changes.



The copy-modify-merge solution, cont.

- ▶ In response, the user needs to *merge* any new changes from the repository into their own working copy before they can commit.
- ▶ The user may need to resolve *conflicts* in the process of merging.



Strategies (Versioning Models)

- ▶ **Lock-Modify-Unlock (Pessimistic Model)**

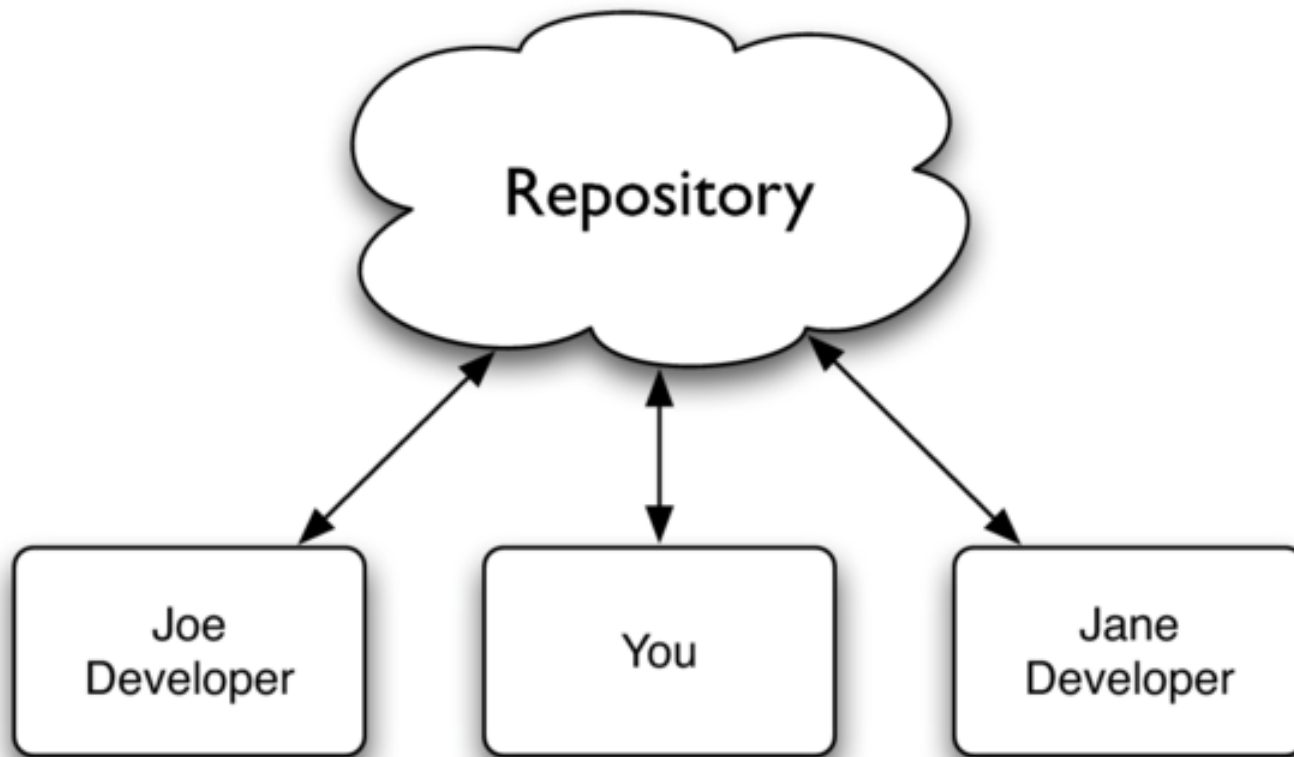
- ▶ Only one person is allowed to change a file at a time.
- ▶ Problem: over restrictive, inhibit productivity.

- ▶ **Copy-Modify-Merge (Optimistic Model)**

- ▶ Each user creates and edits a personal working copy, and finally copies are merged into a new version.
- ▶ Most current version control systems use this mechanism.
- ▶ Problem: some files are hard to merge, such as files of binary formats.

Repository Model

- ▶ Centralized (Client-Server) Repository Model

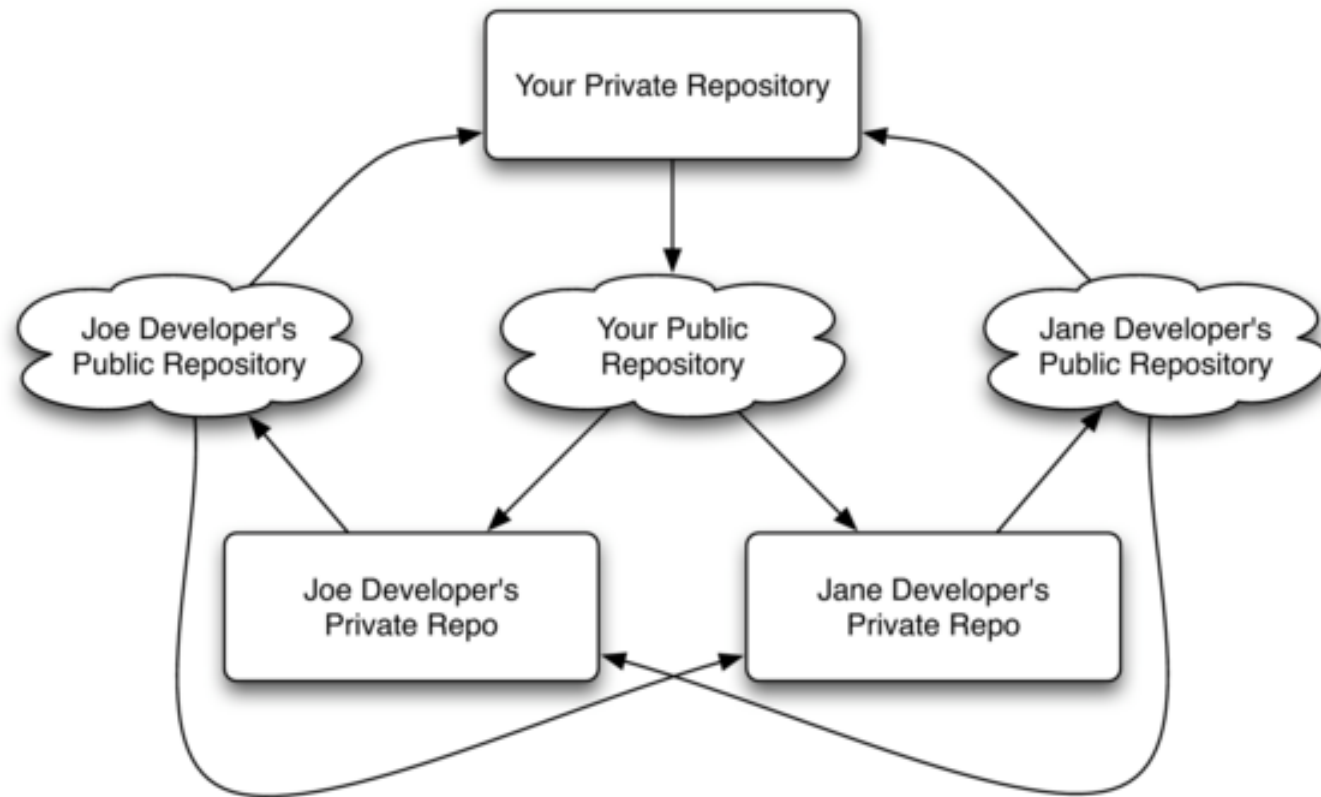


Repository Model

- ▶ **Centralized (Client-Server) Repository Model**
 - ▶ There is a central repository (server) that holds all of the versioned data.
 - ▶ Users (clients) change their working copies, and update the repository to incorporate those changes.
 - ▶ Users have to maintain a network connection to the central repository in order to track the changes that they make.

Repository Model

► Decentralized Repository Model



Repository Model

▶ Decentralized Repository Model

- ▶ Each developer has their own copy of the repository that includes complete history.
- ▶ Anyone can be a server.
- ▶ Each developer has a public repository and a private repository
 - ▶ Public repository is to share changes with other developers
 - ▶ Private repository is to manage drafts that they don't want to publish
- ▶ Network is not involved in most operations.
- ▶ Allows users to work productively even when not connected to a network.

Version Control Systems

- ▶ **Centralized**
 - ▶ Open-source
 - ▶ CVS, **Subversion**
 - ▶ Commercial
 - ▶ Visual SourceSafe, Perforce
- ▶ **Decentralized**
 - ▶ Open-source
 - ▶ Monotone, **Git**
 - ▶ Commercial
 - ▶ TeamWare, BitKeeper

Important Operations of Version Control

▶ Basic Operations

- ▶ Add
- ▶ Check in
- ▶ Check out
- ▶ Update
- ▶ Head
- ▶ Revert

▶ Advanced Operations

- ▶ Branch
- ▶ Merge
- ▶ Tag
- ▶ Diff / Delta

Basic Operations

▶ Add

- ▶ Put a file into the repository for the first time, i.e. begin tracking it with Version Control.

▶ Check out

- ▶ Download a file from the repository. Checking out a repository creates a “working copy” of it on your local machine.
- ▶ The working copy becomes editable (Pessimistic Model only).

▶ Check in (Commit)

- ▶ Upload a file to the repository (if it has changed). The file gets a new revision number (per file v.s. per repository).
- ▶ The working copy becomes read only (Pessimistic Model only).

Basic Operations

- ▶ **Update**

- ▶ Refresh the local or working copy with any changes since checkout.

- ▶ **Head**

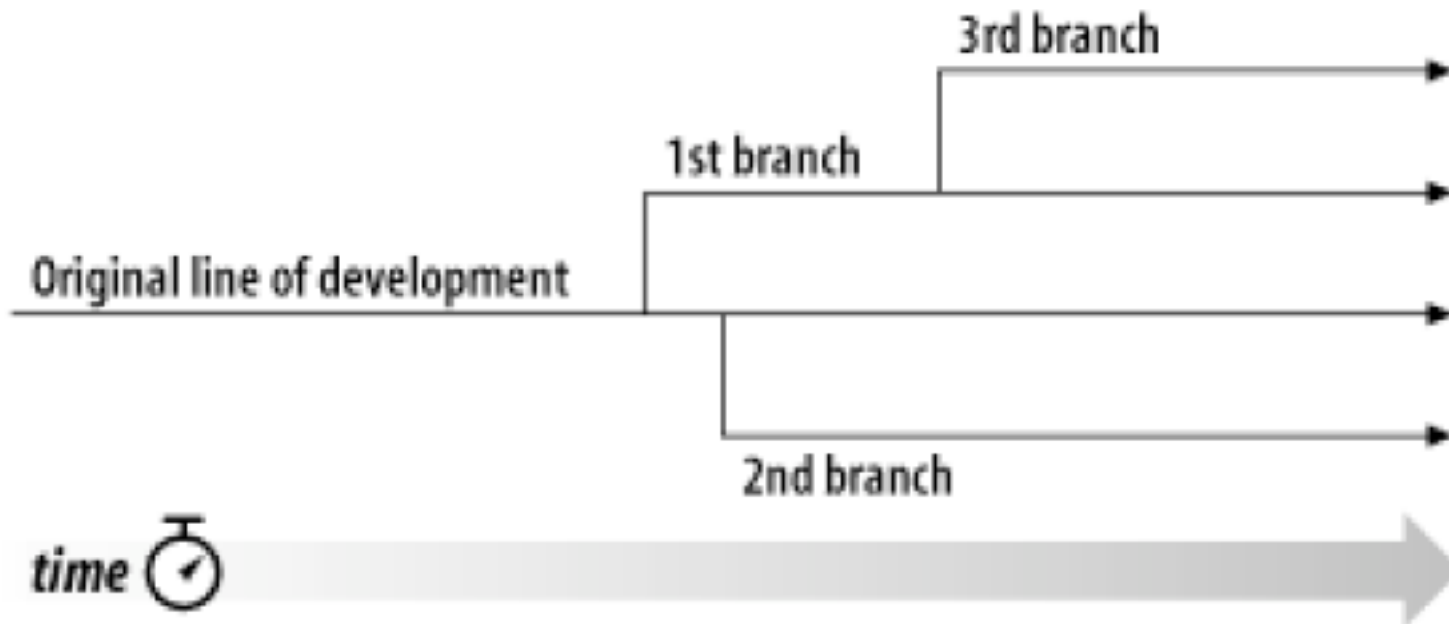
- ▶ The latest revision in the repository (the most recent commit).

- ▶ **Revert**

- ▶ Throw away your local changes and reload the latest version from the repository.

Branch

- ▶ Lines of development that exist independently and share a common history.
- ▶ Each branch generates its own history.



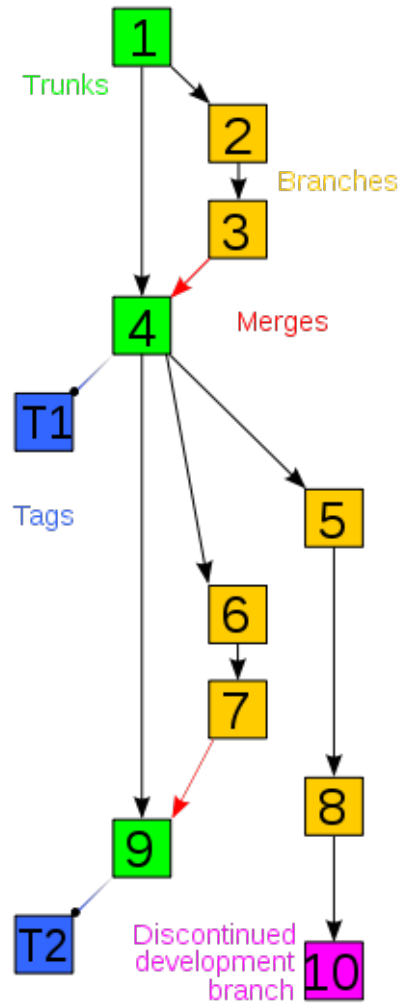
Merge

- ▶ To take some changes which were done to one branch and apply them to another branch.
- ▶ Can cause conflicts
 - ▶ Require user intervention
- ▶ Other issues
 - ▶ Merging histories
 - ▶ Repeated merge

Tag

- ▶ A “snapshot” of a project in time.
- ▶ A name that is given to a particular commit; the name that you want that particular commit to be represented by.
- ▶ Makes it easy to check out a release or configuration.

Branch, Merge, and Tag



Diff/Delta

- ▶ Diff/Delta: a specific modification to a document under version control.
- ▶ Most version control systems store diffs rather than full copies of the file to save disk space (Delta Compression).

Best Practices

- ▶ Explain your checkins completely.
- ▶ Keep in touch with the repository (update often).
- ▶ Checkin your work to the repository as often as you can without breaking the build (commit often).
- ▶ Review the merge before you commit.
- ▶ Use checkout and revert with caution.

Reference

- ▶ Pragmatic Version Control Using Git by Travis Swicegood
- ▶ Wikipedia:Revision control.
- ▶ Version Control with Subversion by Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato.
 - ▶ Many diagrams of this lecture are from this book.
 - ▶ The book is free online: <http://subversion.apache.org>