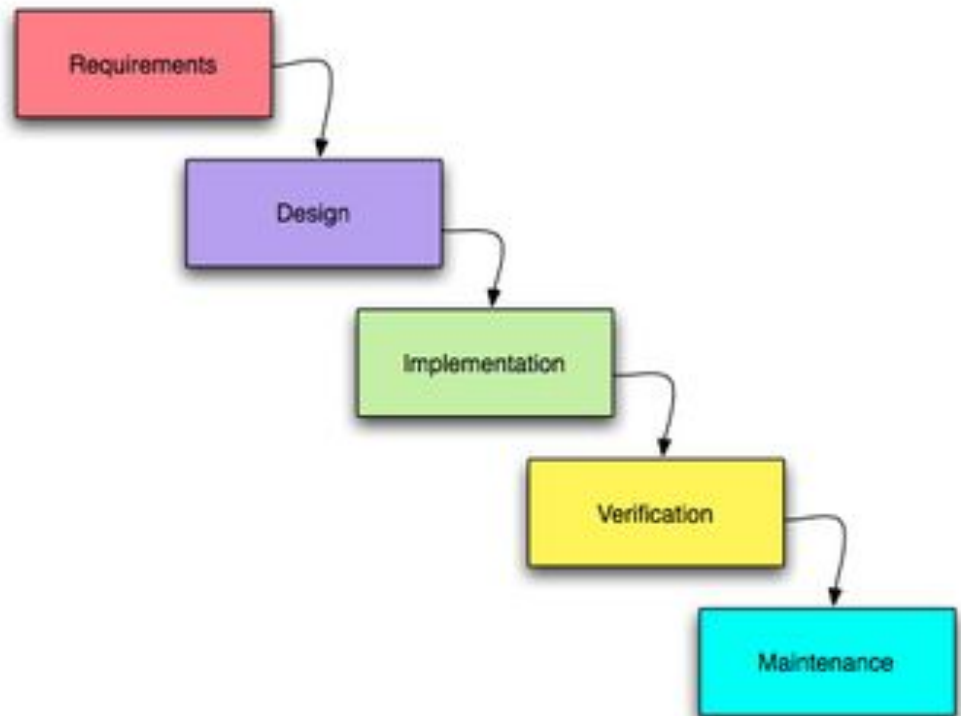


Introduction to Version Control

The stages of developing a software application

- Requirements Analysis
- High-level Design
- [Plan (SE2800)]
- Low-level Design
- *Implementation*
- Unit Test (SE2832)
- *Integration*
- *System Test (SE3800)*
- Deploy (SDL)
- Maintain (SDL)

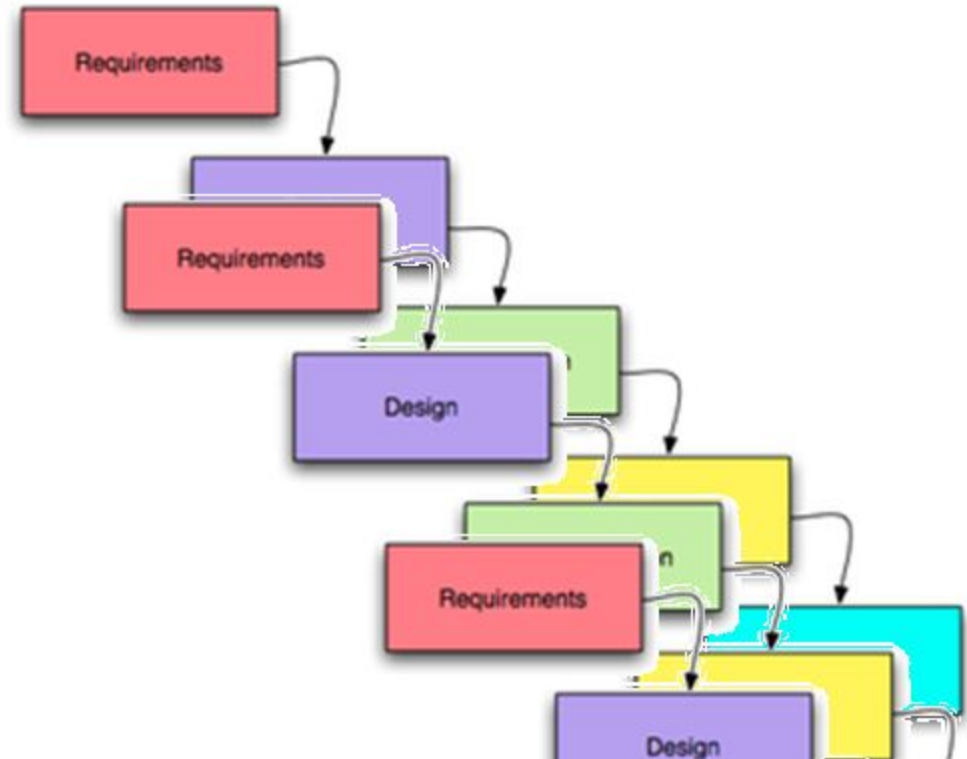


In many cases, multiple projects run concurrently

New features are typically
being added to the next
version

While at the same time,
defects need to be
corrected in existing
releases

You will do this more in SDL



On a single project, a team of software engineers work together toward a release

All team members work on the same code and develop their respective parts

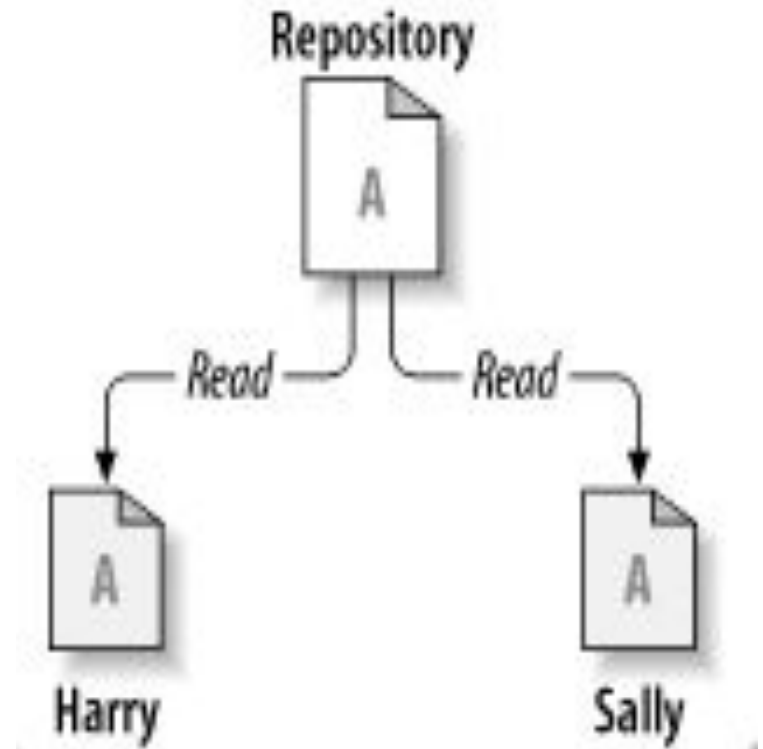
An engineer may be responsible for an entire class

Or just a few methods within a particular class



Files have to be shared among developers on a team project

Shared files can be kept in some type of **Repository** where they can be accessed and worked on by multiple individuals

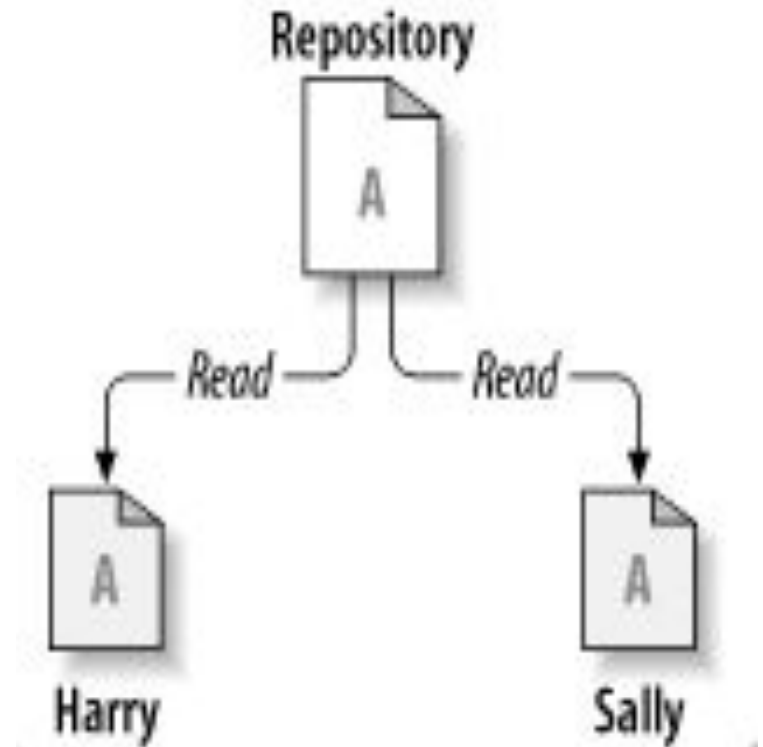


Harry and Sally get their own respective local **Working Copies** of the Master File in the Repository

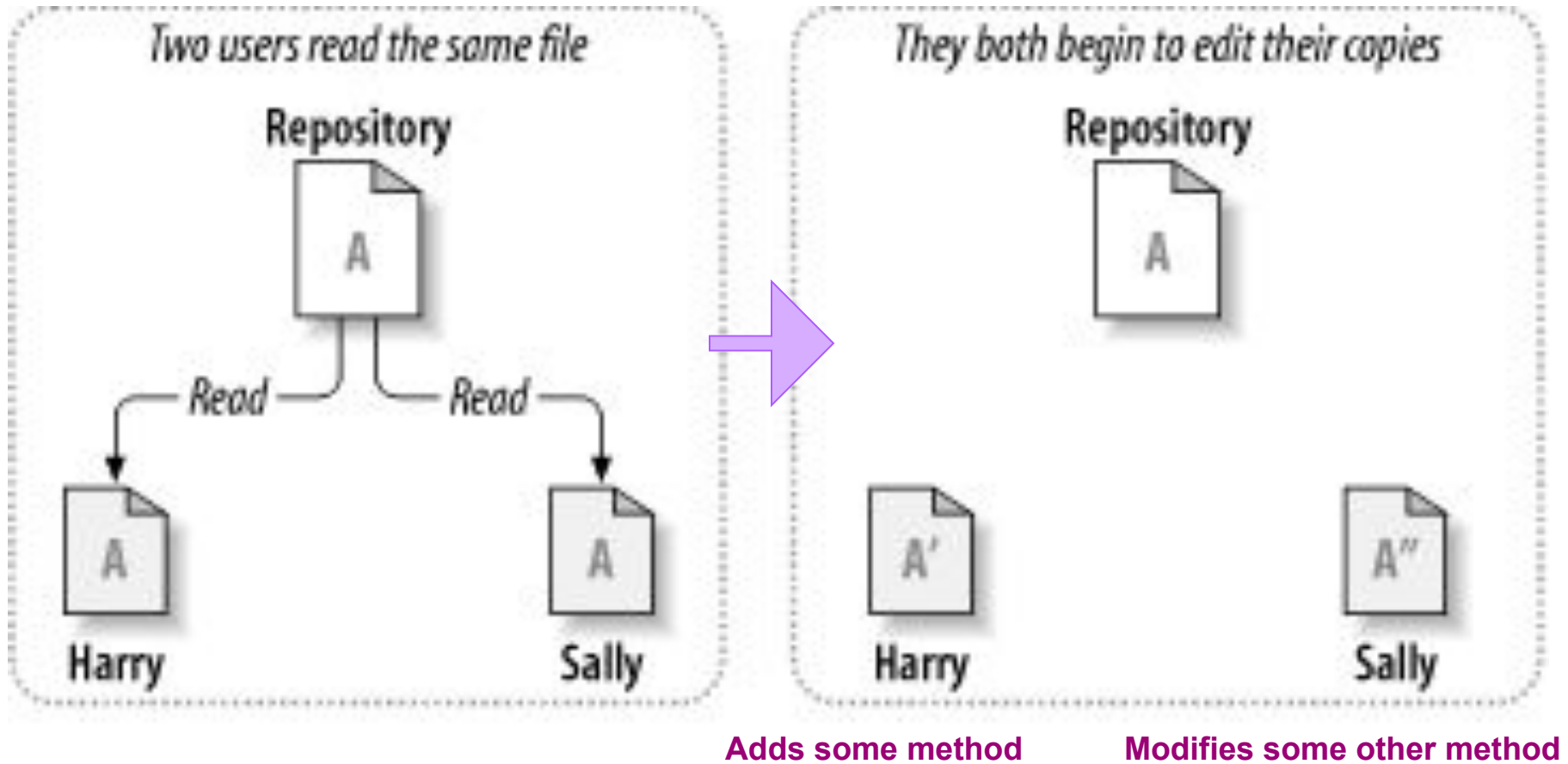
What might you use for a Repository?

Would any of these work?

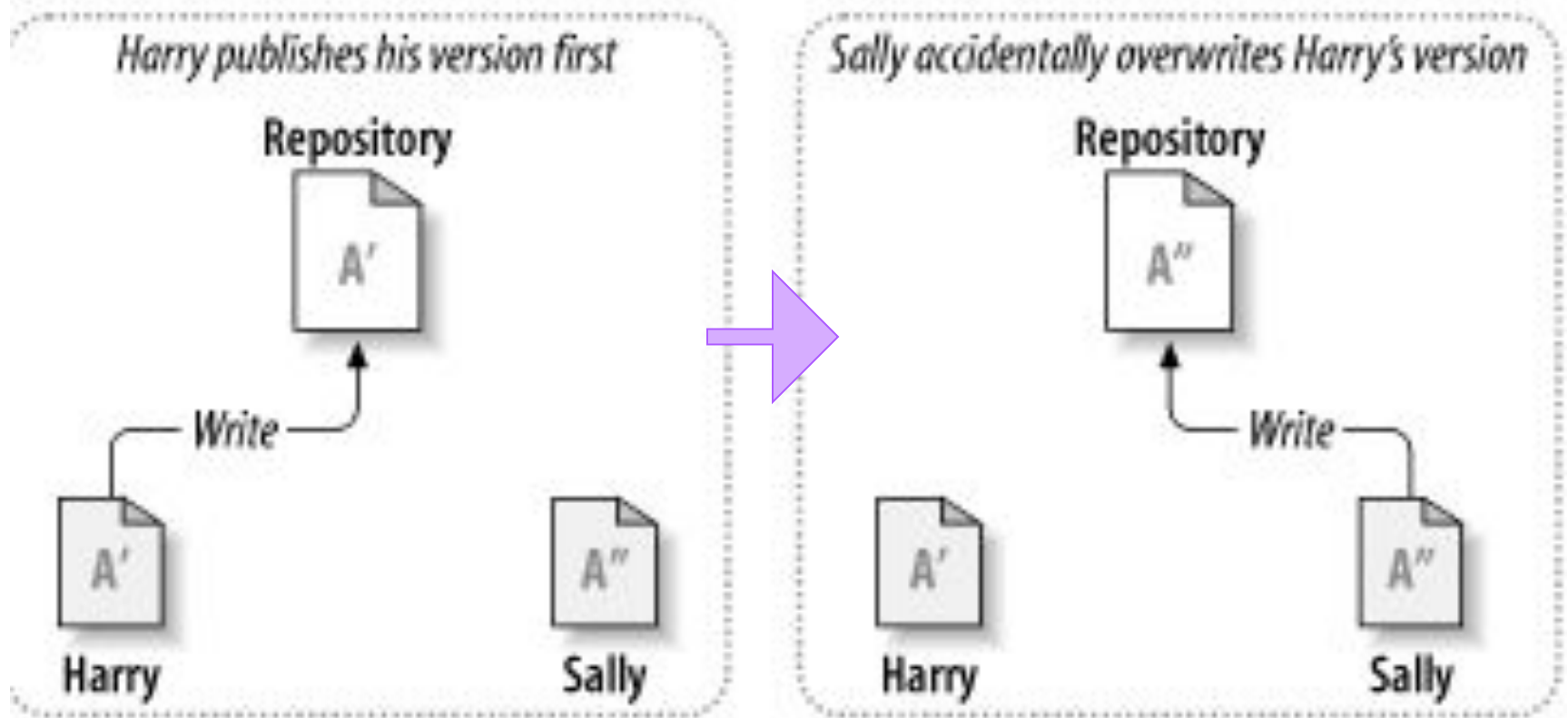
- USB/SD drive
- Private network drive
 - E.g. shared “X:” drive
- Cloud drive
 - Google drive
 - Dropbox
 - iCloud



Sharing files can lead to big problems...

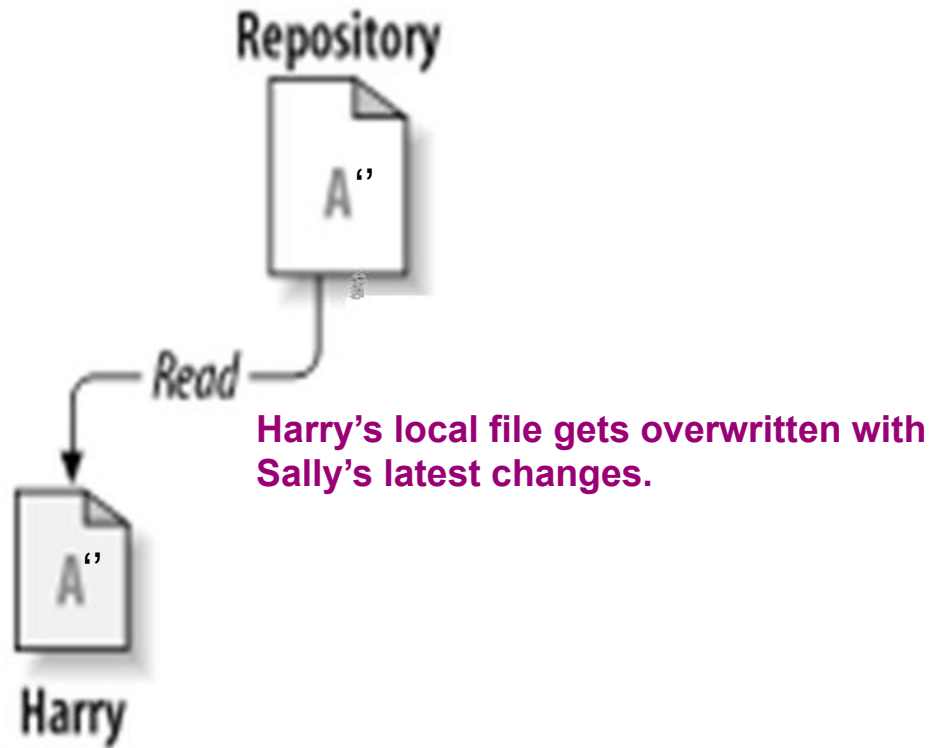


The problem to avoid:



Harry's changes are still held locally...

Harry's changes are lost



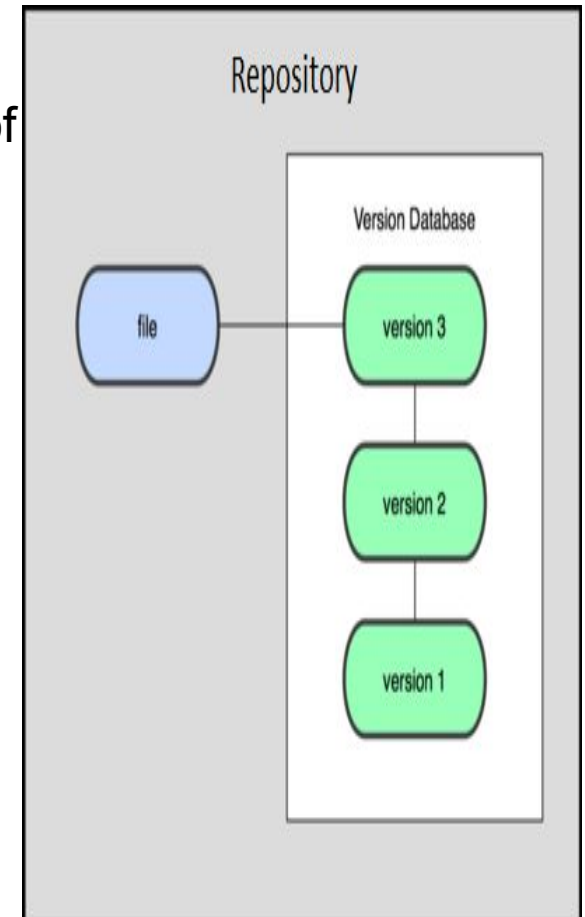
The Issue: Effective management of software artifacts, especially code

Some acronyms:

- SCM – Source Code Management
- SCCM – Source Code Configuration Management
- RCS – Revision Control System
- VCS – Version Control System
- DVCS - Distributed Version Control Systems

Revision/Version History might help

- ...but USB/SD, private network drives do not maintain revision history
- Cloud drives (e.g. **Google Drive**, **Dropbox**, **iCloud/Time Machine**) maintain a (limited) history of the various revisions of a file
 - **Google Drive: keeps last 100 revisions or last 30 days**
 - Supported in “Classic” mode only – going away?
- **IF** you detect a collision, you can manually merge the differences and resynchronize your work
 - Might be workable for two people – what about 5 or 10?



Any other disadvantages to Google Drive/Dropbox/iCloud

- Security?
- Support (bug fixes and new features)?
- Ownership of stored information?
- Cost?
- Ease of Use?
- Acceptance?

Special-purpose Revision Control Systems have been around a lot longer than Google Drive or Dropbox

- SCCS, 1972 (IBM)
- RCS, 1982 (Unix)
 - SourceSafe 1995 (Microsoft)
- CVS, 1986 (multi-platform)
- SVN, 2000 (multi-platform)
- Git and Mercurial, 2005 (multi-platform)

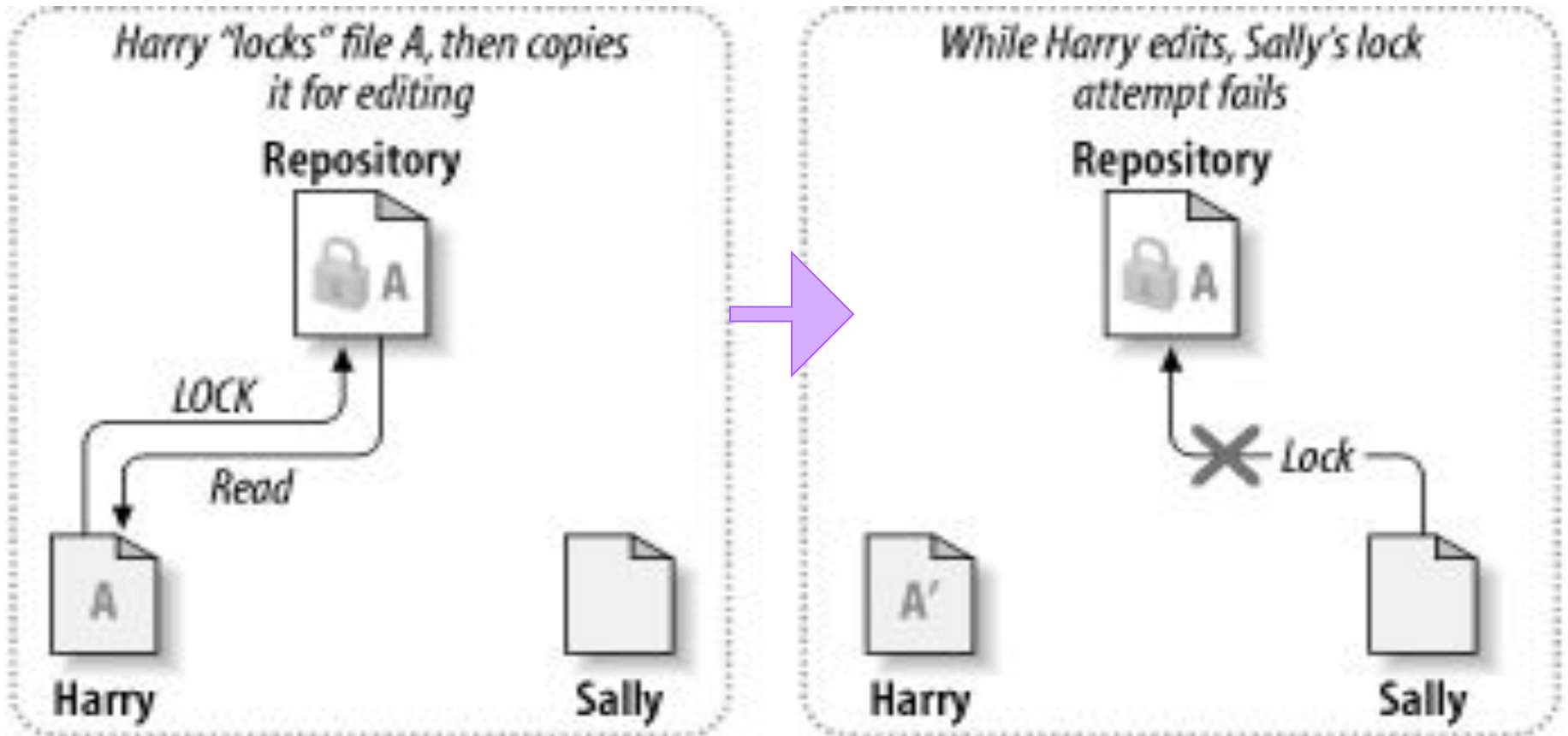
The way these work has evolved over the years

Features of version control systems

- All changes to a file are tracked
 - Version histories show **who, what, when**
- Changes can be reverted (rewound to any earlier version)
- Changes between revisions are easy to identify

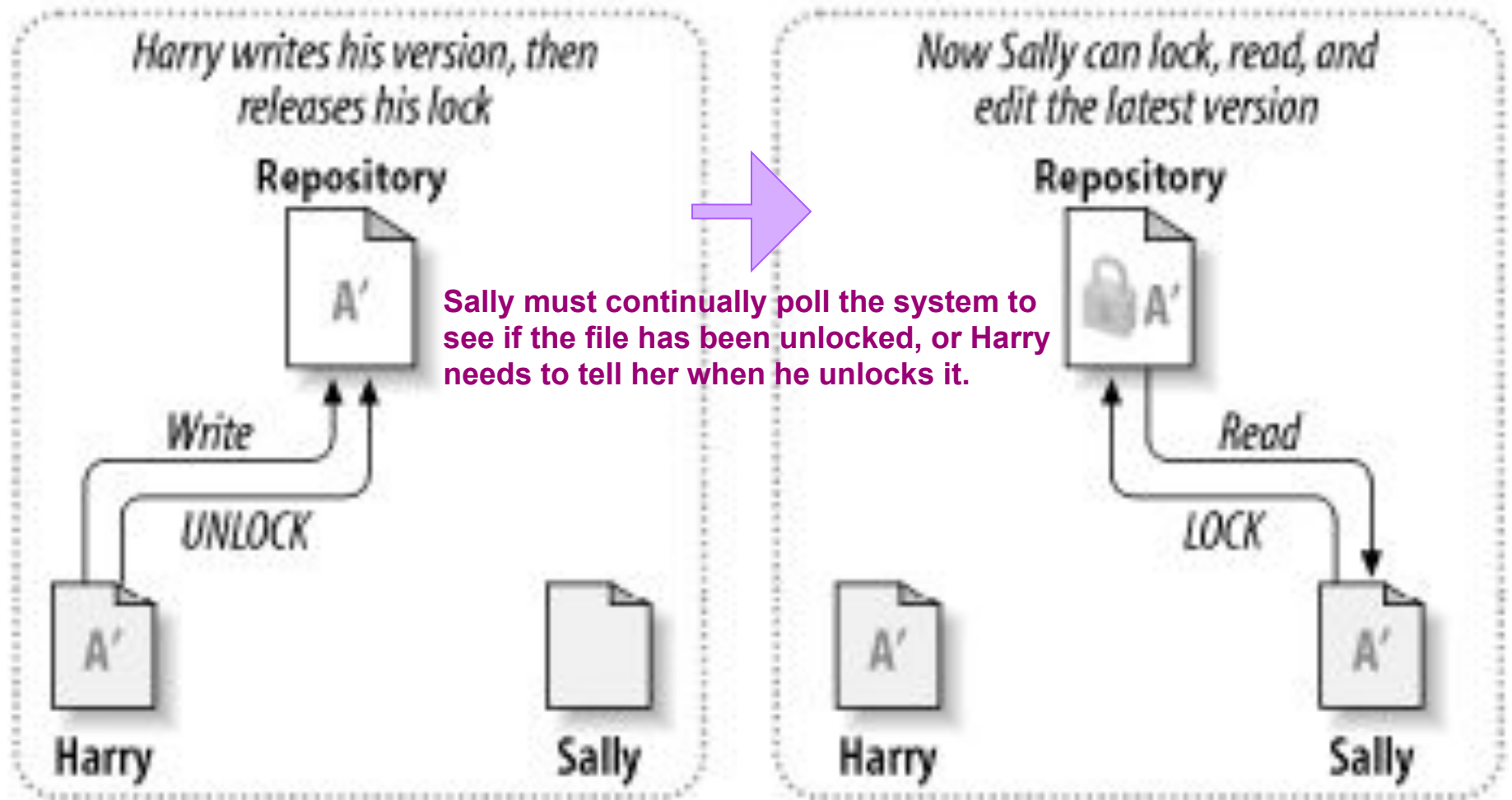
The Lock-Modify-Unlock solution

Early systems followed this model
(RCS, SourceSafe)



Sally must wait until Harry releases the lock, although she can retrieve a "readonly" version of the file in the meantime.

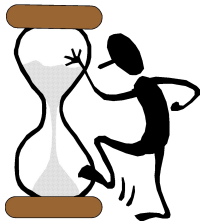
Lock-Modify-Unlock only allows a single user to edit the file at a time



Lock-Modify-Unlock has certain drawbacks:

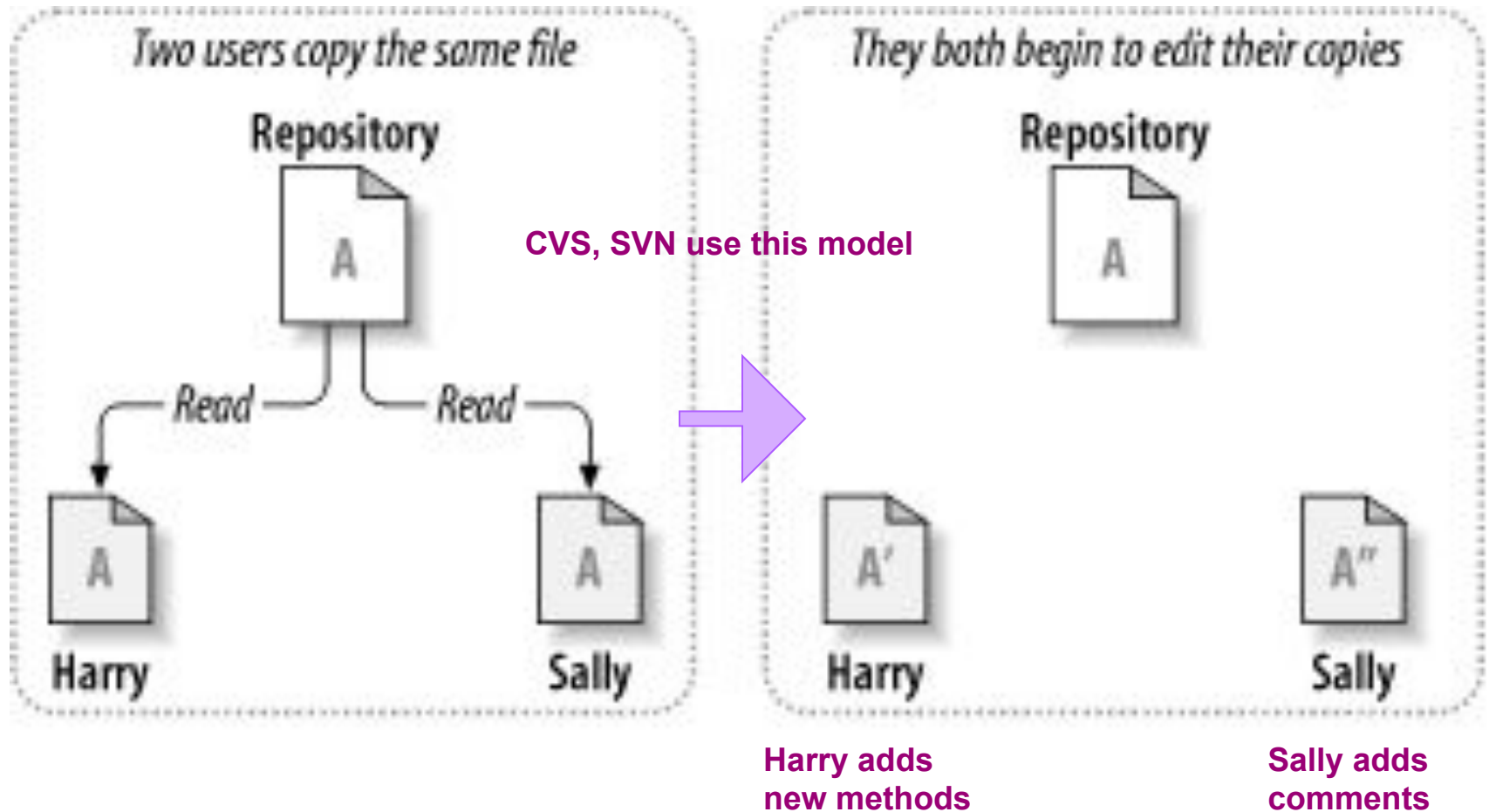
Harry has to remember to release his lock before Sally (or anyone else) can acquire the lock in order to edit

Sally has to wait for Harry to finish before editing (editing must be done sequentially)

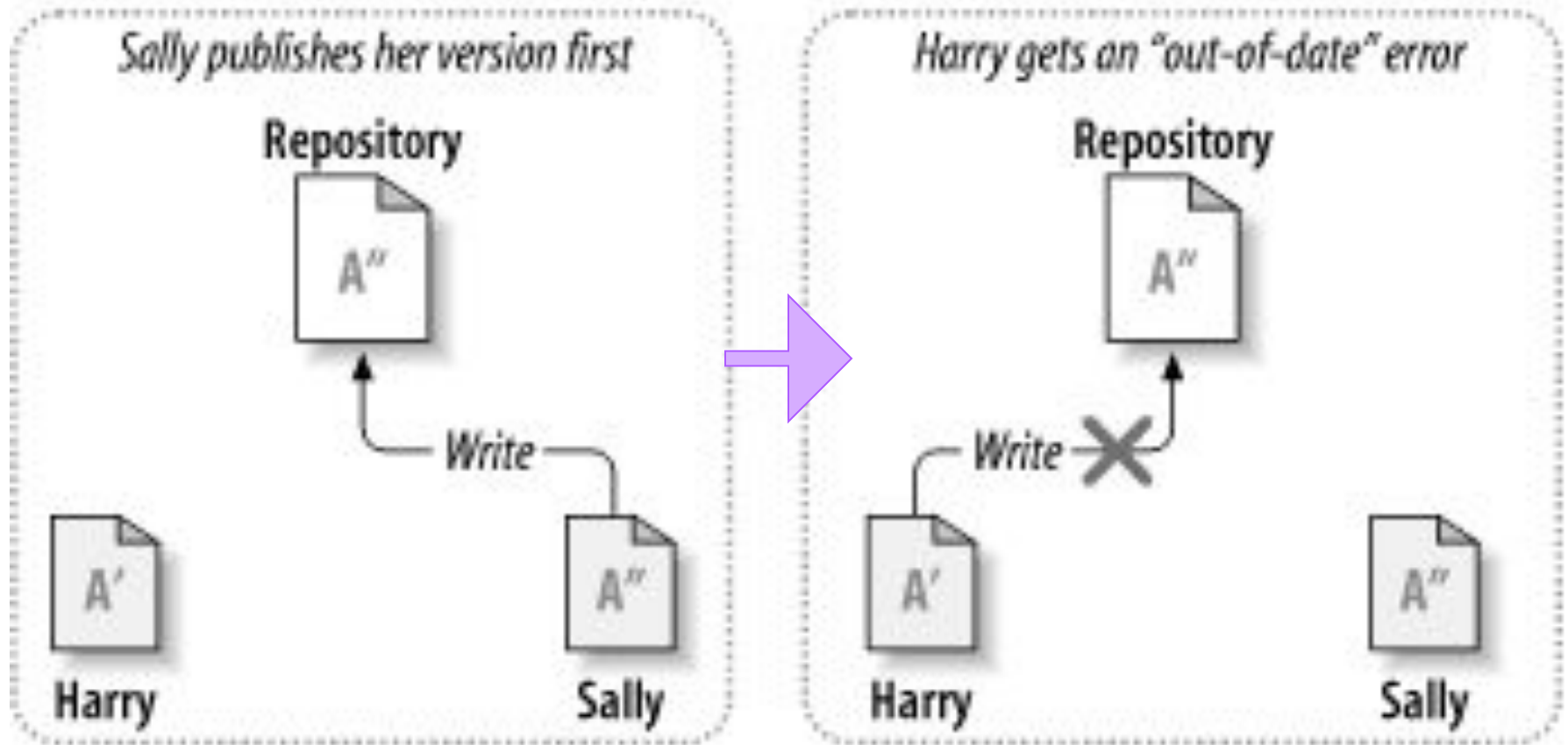


- Harry might be adding some new methods (takes a long time)
- Harry might forget to release the lock before going home (or on vacation!)

The Copy-Modify-Merge solution allows concurrent editing

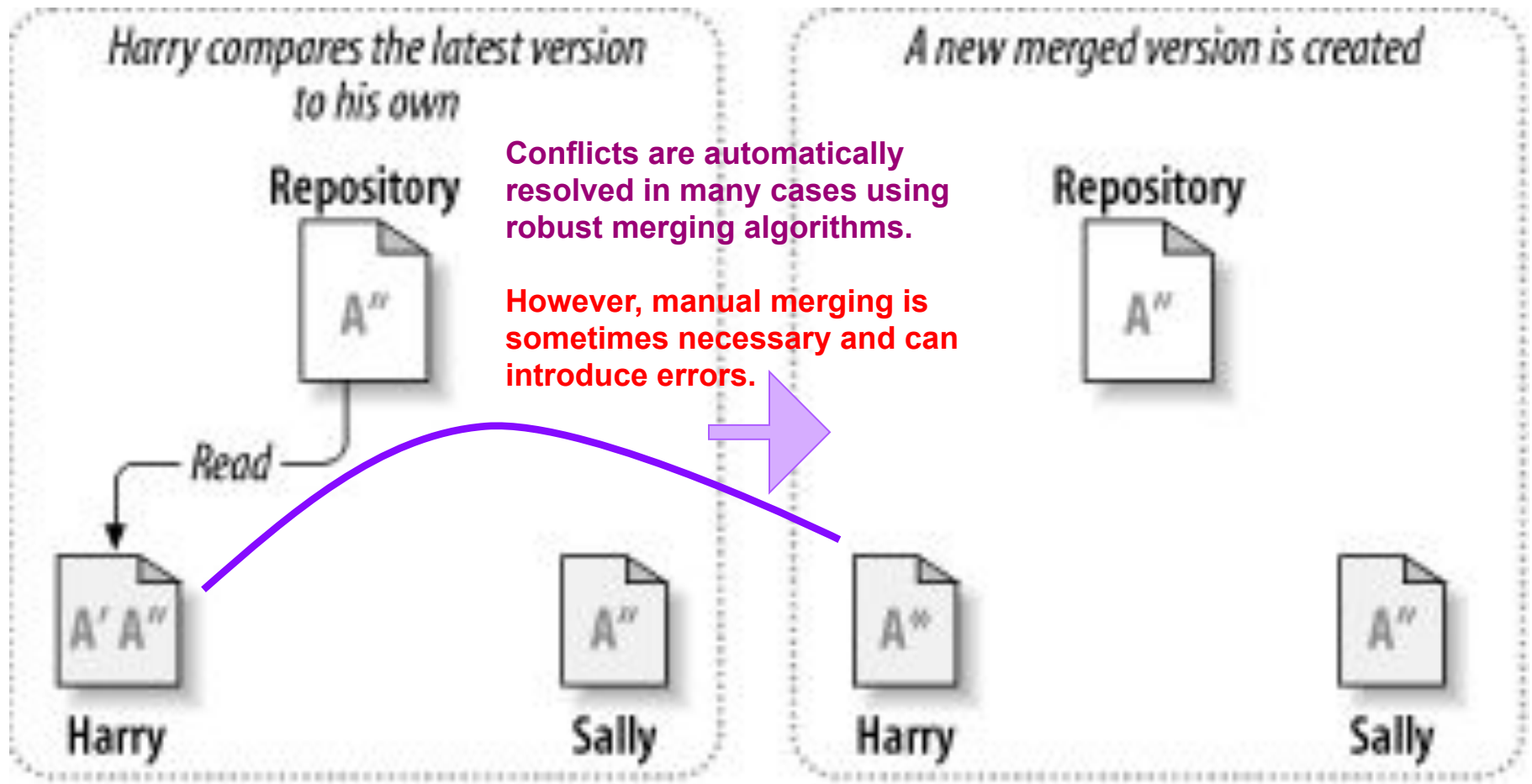


The Repository recognizes conflicts

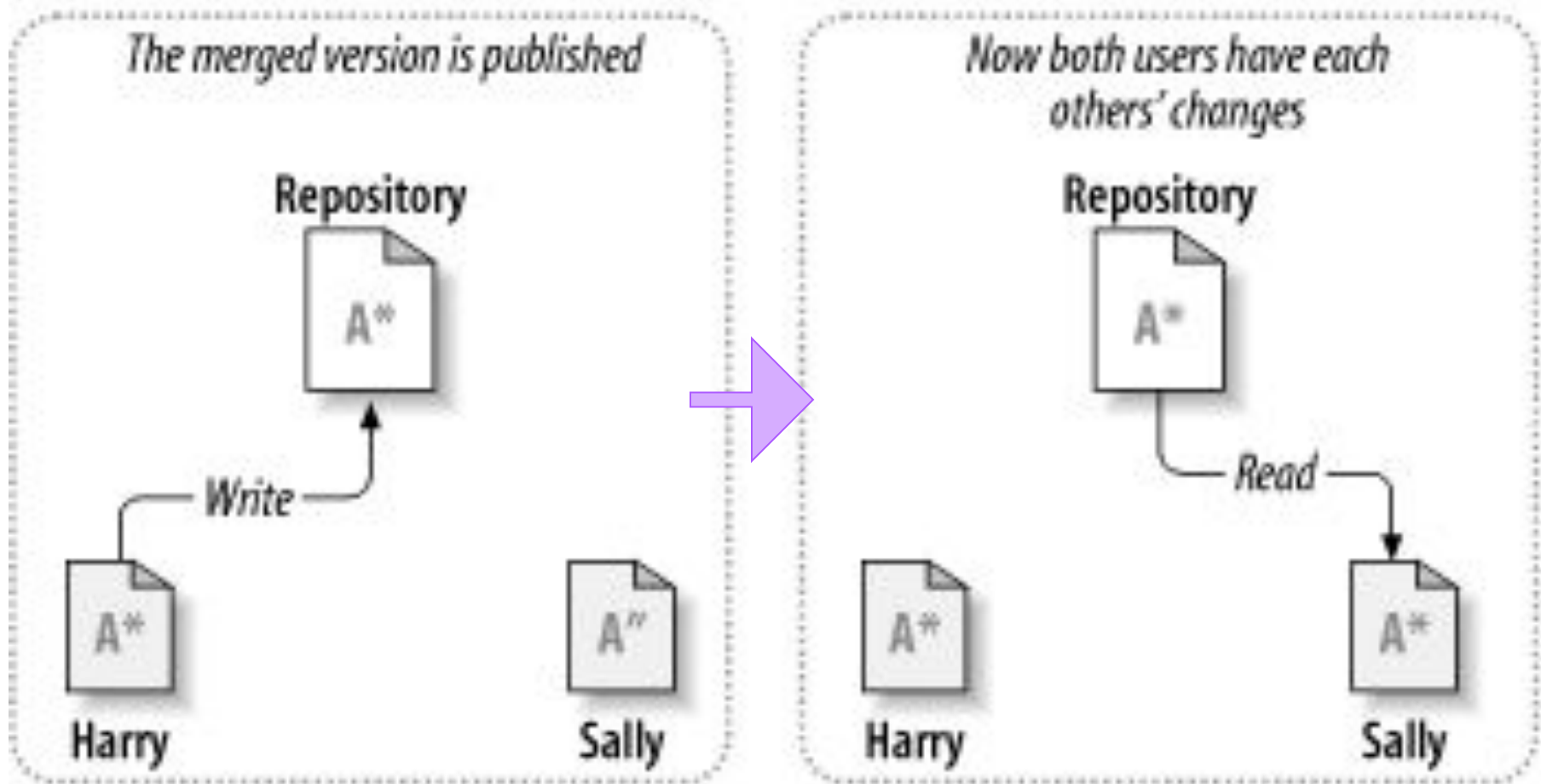


Harry is prevented from writing

Conflicting versions of the files are merged



The Repository keeps both users synchronized



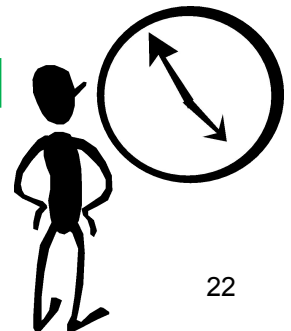
Copy-Modify-Merge is generally easy to use and manage

Users can work in parallel

Most of the time, concurrent changes don't overlap



- People generally don't edit exactly the same code simultaneously, so merging is done automatically in many cases
- Amount of time spent resolving conflicts is nearly always less than the time that would be spent waiting for a lock to be released



Is Lock-Modify-Unlock ever needed anymore?

When two or more people need to work on the same file, the simultaneous changes may not be mergable in all cases:

- MS Office documents
- Image documents
- Other binary files



Subversion is an open-source version control system that is available and still supported for many platforms

- Windows
- Mac
- Linux



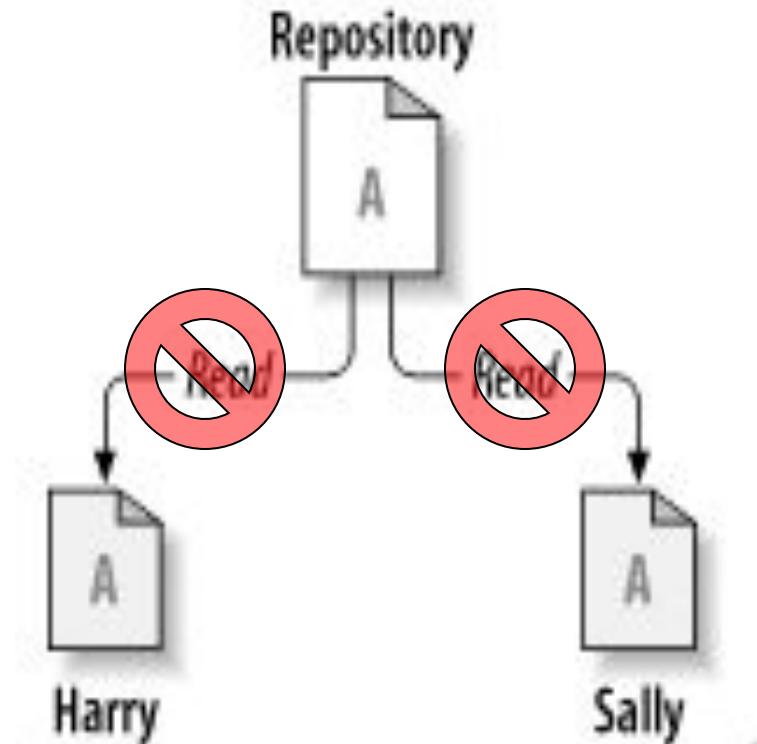
Many current open-source projects use Subversion to maintain source code control

- <http://subversion.tigris.org/>
- Latest release 8/2014
- MSOE still maintains Subversion on a linux server

Subversion's main drawback is the centralized nature of the Repository

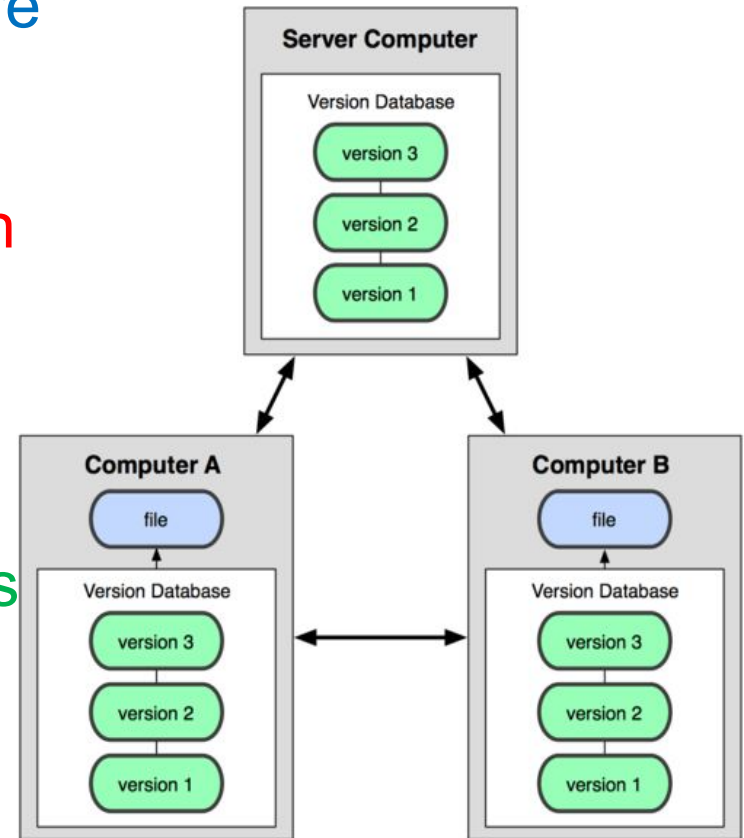
A repository is typically hosted on a server.

- Version history is on the server, not on your local PC
- If the server or network become unavailable, everyone gets stuck with the working copy they have.



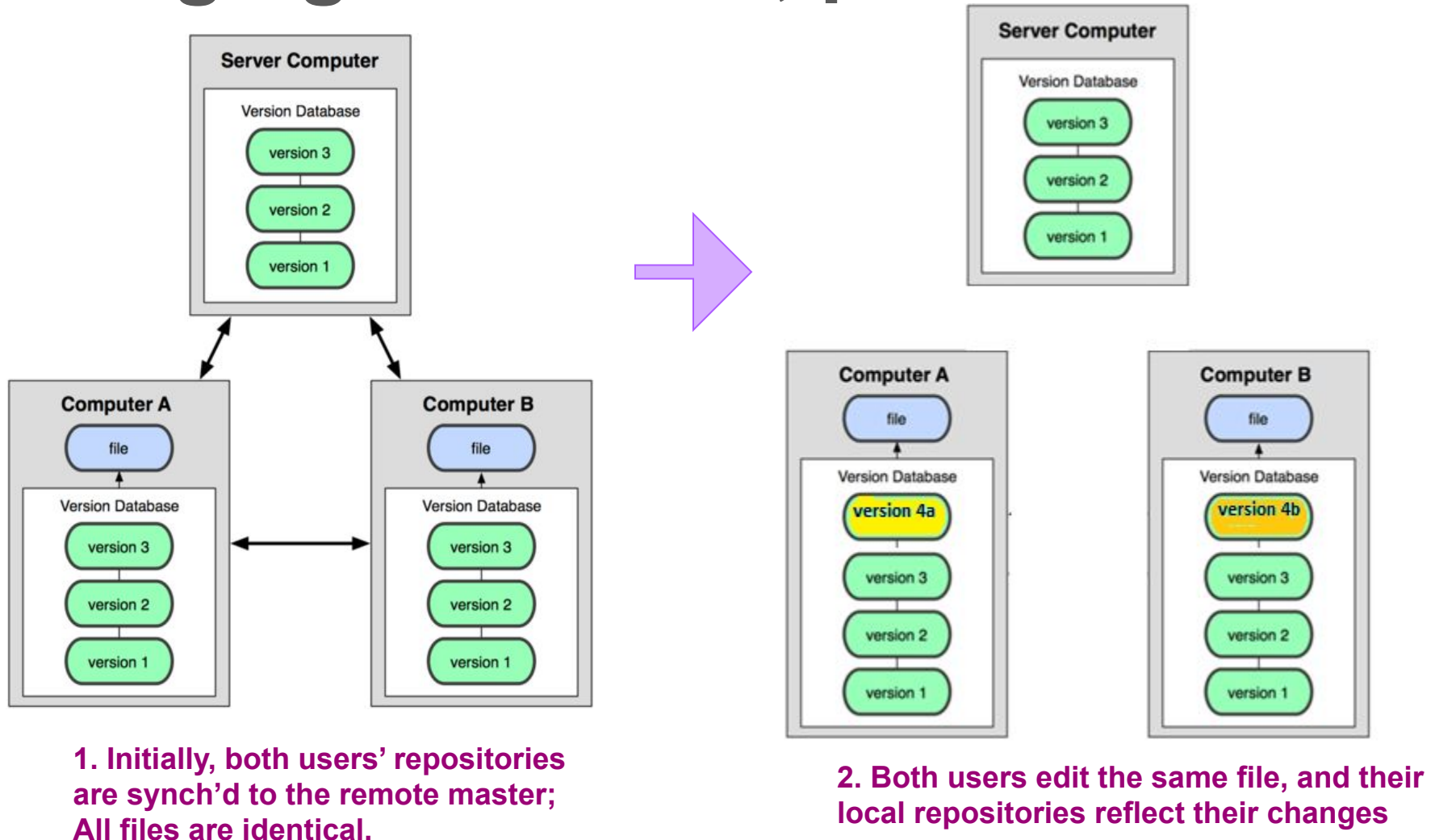
Distributed Version Control

- A “remote master” repository is typically hosted on a server.
- Clones of the remote repository are maintained on each user's computer
- If the server goes down, users can continue sharing code (provided they can connect over a network) by synch'ing to each others' repositories
- If the network is unavailable, users can continue reading & writing to their own local repository – synch'ing with others later

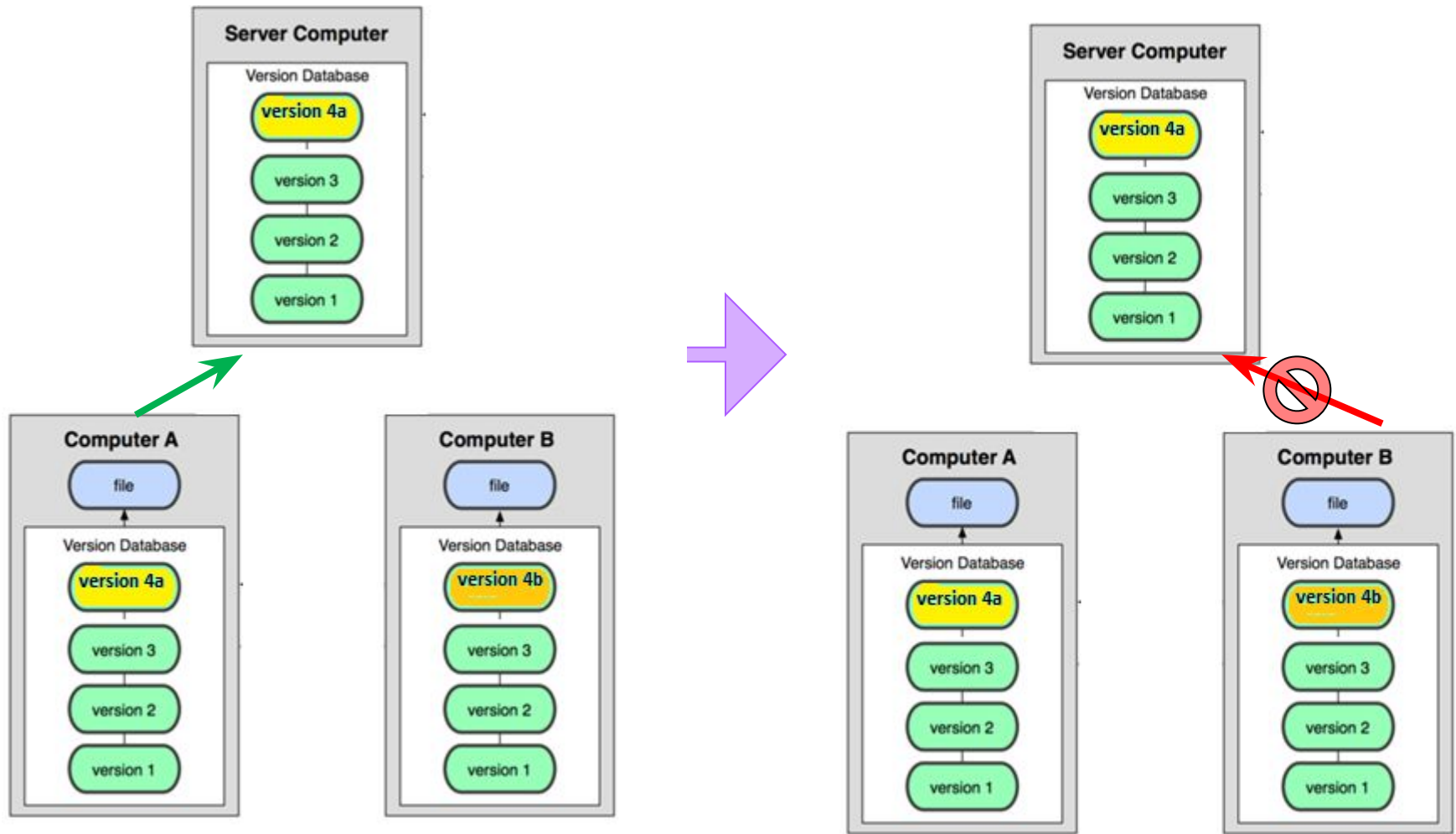


Git and Mercurial use this model

Merging in a DVCS, part 1



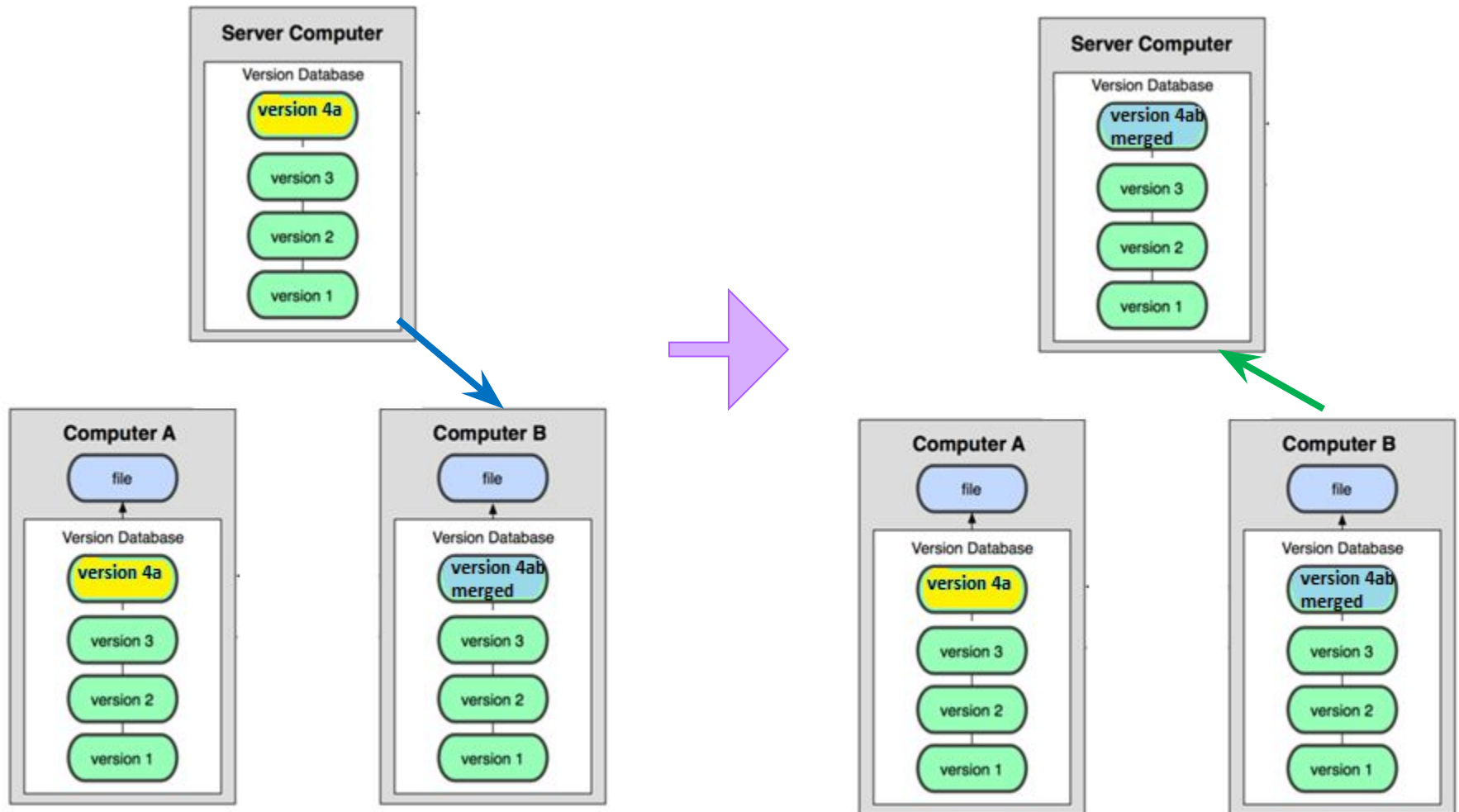
Merging in a DVCS, part 2



3. User A “pushes” her repository to the remote master to sync; this succeeds.

4. User B attempts to “push” his repository to the remote to sync; this fails due to User A’s earlier push. Remote does not change.

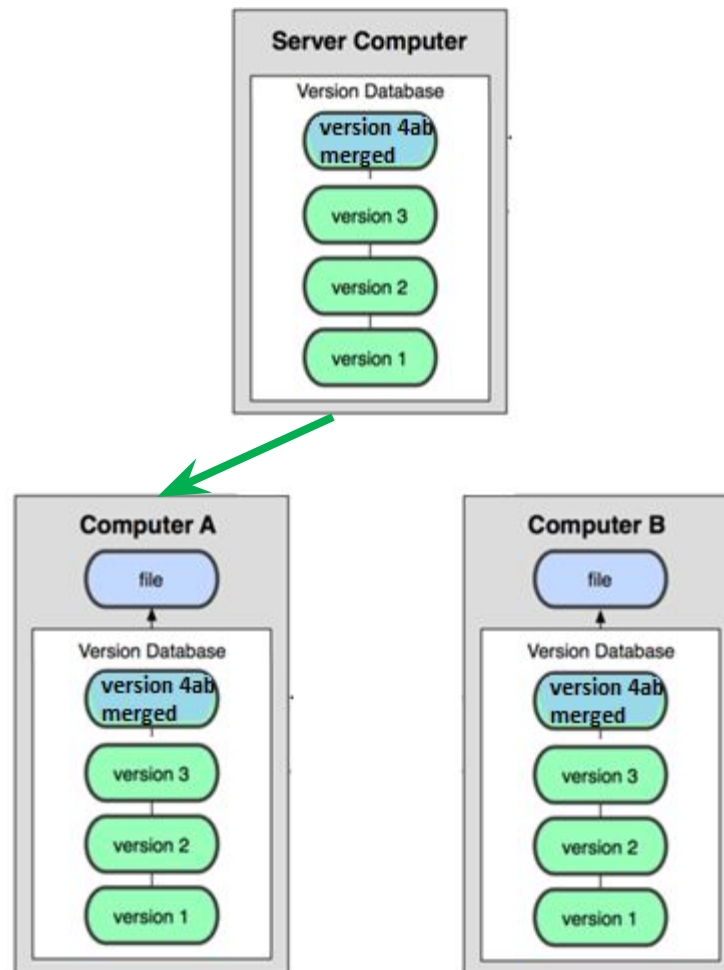
Merging in a DVCS, part 3



5. User B “pulls” User A’s updates from the remote master, merging A and B together. **Merging is automatic with some exceptions.**

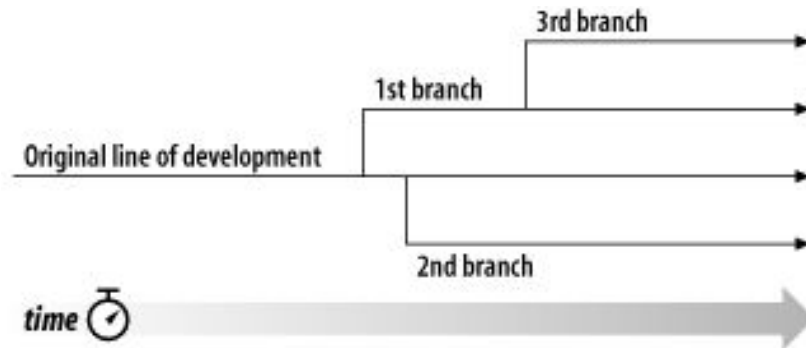
6. User B “pushes” his repository to the remote to sync; this now succeeds

Merging in a DVCS, part 4



7. User B “pulls” from the remote, and everyone is in synch again.

Repositories can also manage and track differences between parallel revisions of a document



- Typically, a software product will undergo parallel development on different **branches** (e.g. for new/future features) while primary development takes place on the main (***master***) branch
- You will learn how to use branching in SE2800; we will not use it in SE2030