CSCB07 - Software Design

Version Control

Based on the slides by Joe Bettridge

Summer 2023

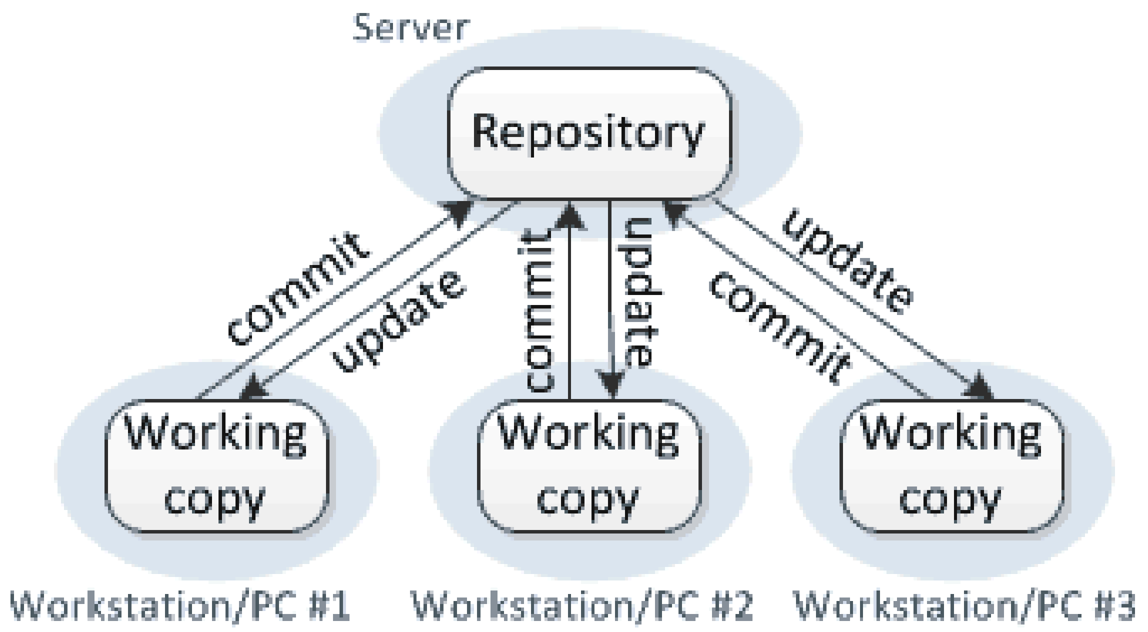
Problems for Developers

1. How do you keep track of your changes?
   - Don’t keep track
   - Save backups periodically
2. How do you decide who has the authority to make changes?
   - Worry about it after the fact
   - Exchange emails / phone calls
3. How do you keep track of code being worked on at home and in office/lab?
   - Copy everything, everytime
   - Try to remember what changed

Solution: Version Control

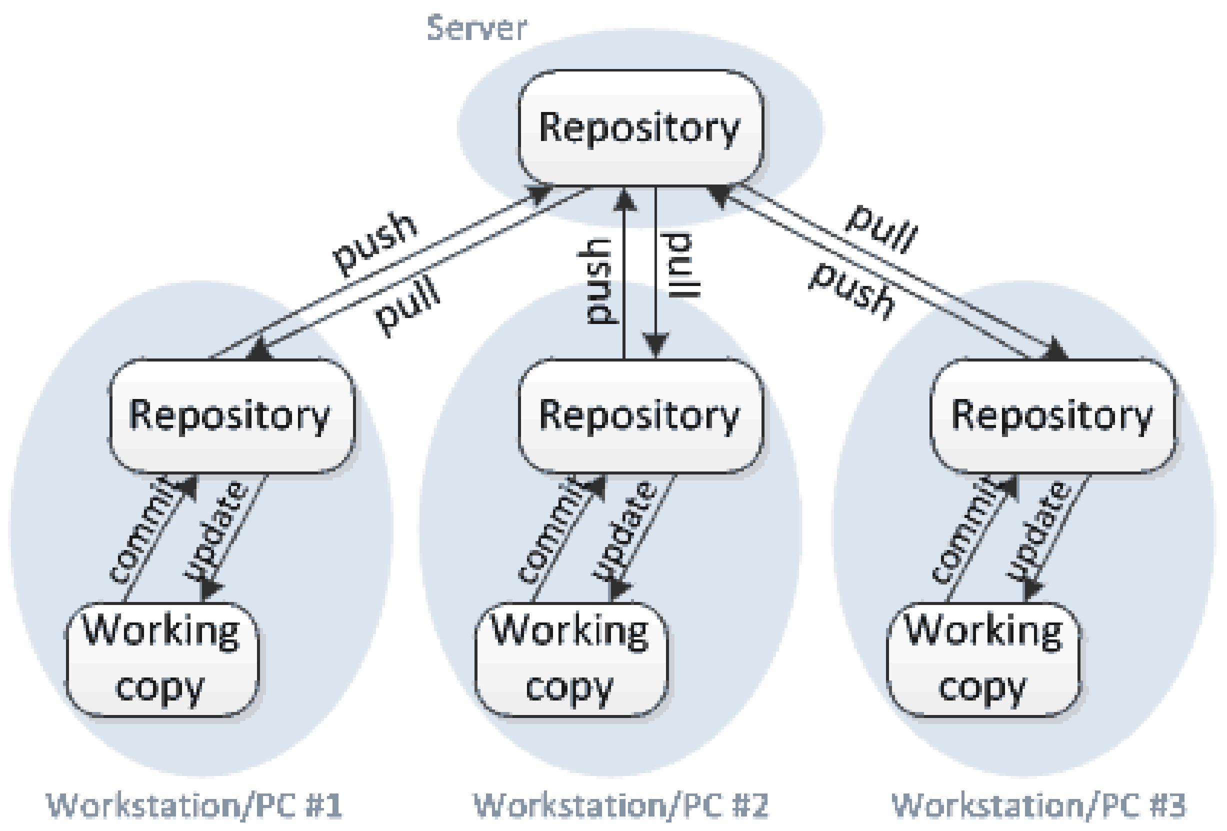
- Two flavours: Centralized (e.g. SVN) and Distributed (e.g. Git)

Centralized Version Control



Source: https://homes.cs.washington.edu/~mernst/advice/version-control.html

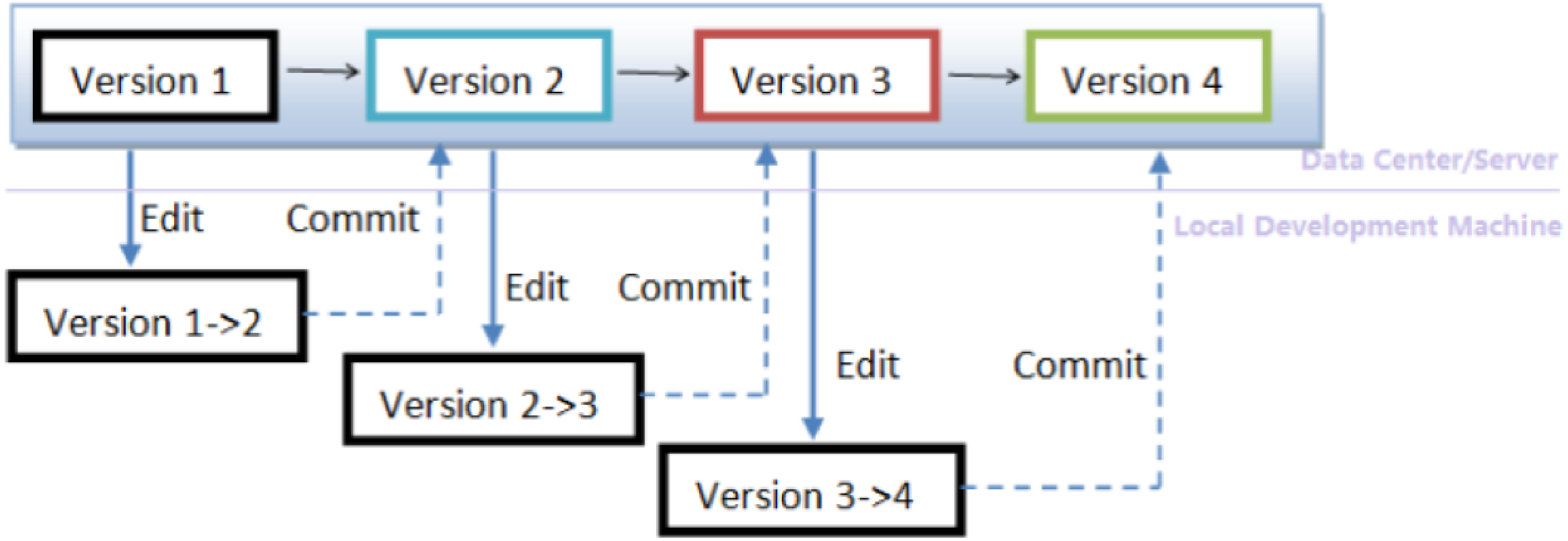
Distributed Version Control



Source: https://homes.cs.washington.edu/~mernst/advice/version-control.html

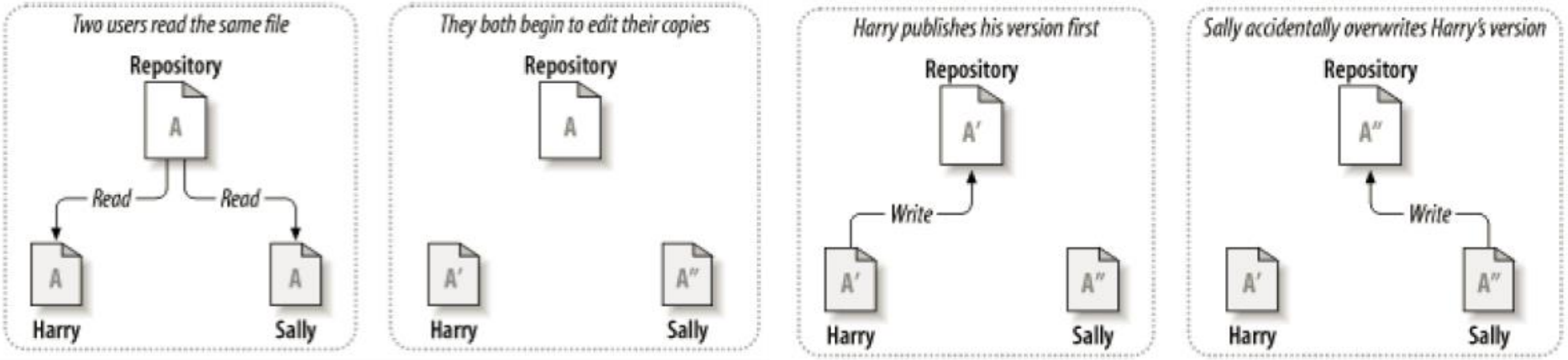
Tracking Changes When Working Alone

- When you get something working, or if you are about to make major changes you may later want to revert, commit
- Tools will allow you to revert to a previous version of the source code (only when commits have occurred)



Version Control – Managing Concurrency

What if two or more people want to edit the same file at the same time?


Two users read the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Harry publishes his version first
Repository
A′
Write
A′
A″
Harry
Sally
Sally accidentally overwrites Harry's version
Repository
A″
Write
A′
A″
Harry
Sally

Source: http://svnbook.red-bean.com/

Version Control – Managing Concurrency

What if two or more people want to edit the same file at the same time?

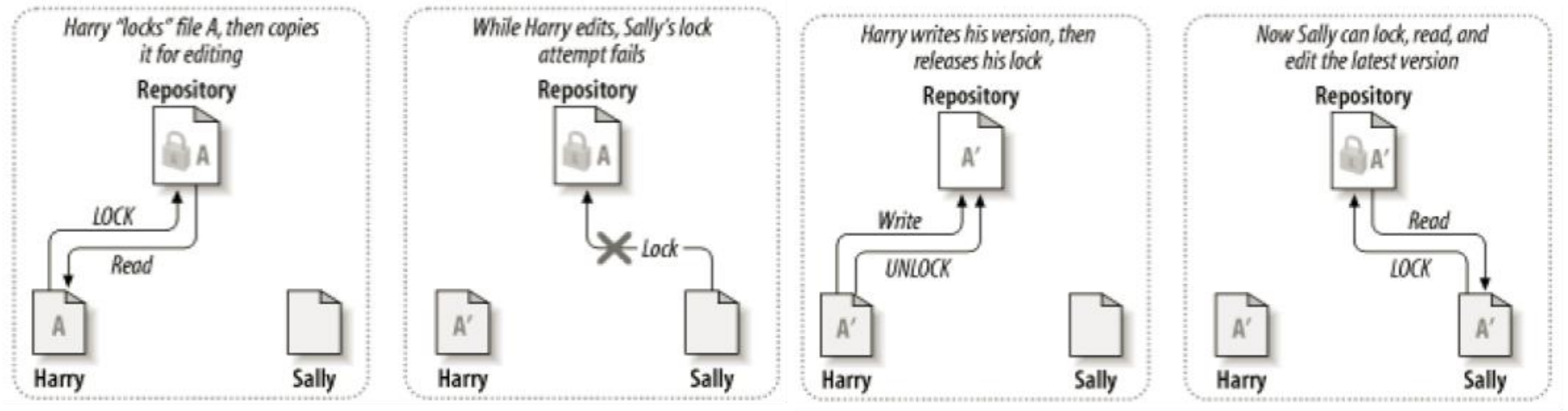
Option 1: Prevent it

- Only allow one writeable copy of each file
- Known as pessimistic concurrency
- E.g. Microsoft Visual SourceSafe, Rational ClearCase

Option 2: Allow it, fix issues afterwards

- Optimistic concurrency
- E.g. SVN, CVS, Perforce

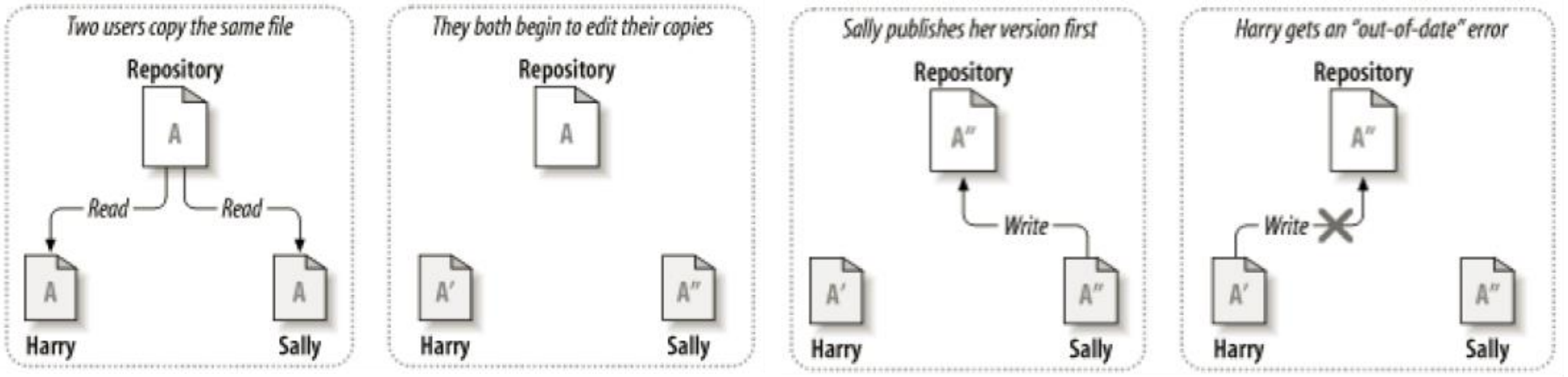
Pessimistic Concurrency



Harry "locks" file A, then copies it for editing
Repository
A
LOCK
Read
A
Harry
Sally
While Harry edits, Sally's lock attempt fails
Repository
A
Lock
A'
Harry
Sally
Harry writes his version, then releases his lock
Repository
A'
Write
UNLOCK
A'
Harry
Sally
Now Sally can lock, read, and edit the latest version
Repository
A'
Read
LOCK
A'
Harry
A'
Sally

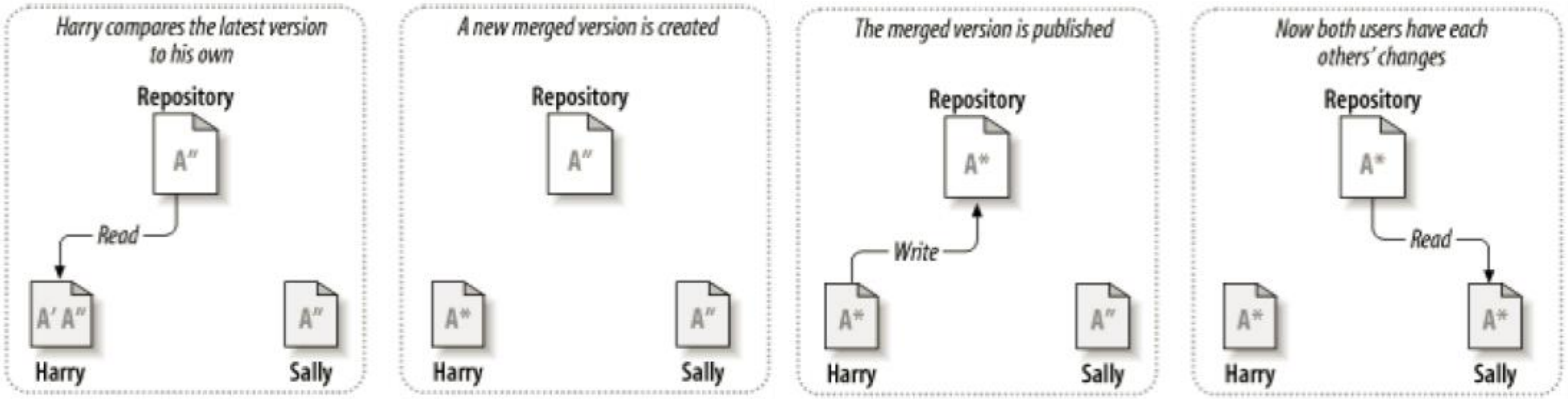
Source: http://svnbook.red-bean.com/

Optimistic Concurrency (1/2)


Two users copy the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Sally publishes her version first
Repository
A″
Write
A′
A″
Harry
Sally
Harry gets an "out-of-date" error
Repository
A″
Write
A′
A″
Harry
Sally

Source: http://svnbook.red-bean.com/

Optimistic Concurrency (2/2)


Harry compares the latest version to his own
Repository
A″
Read
A′ A″
Harry
A″
Sally
A new merged version is created
Repository
A″
A*
Harry
A″
Sally
The merged version is published
Repository
A*
Write
A*
Harry
A″
Sally
Now both users have each others' changes
Repository
A*
Read
A*
Harry
A*
Sally

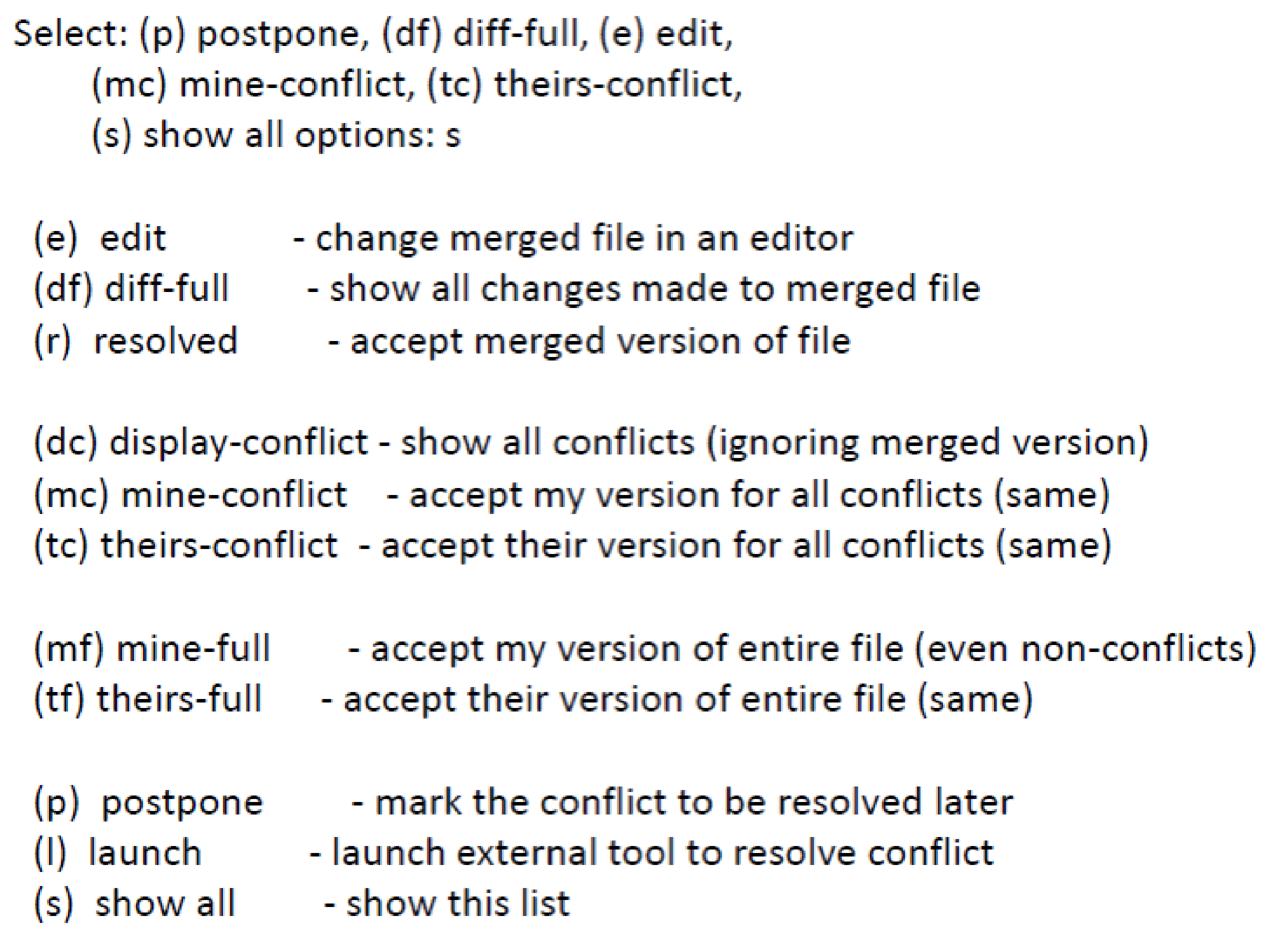
Source: http://svnbook.red-bean.com/

Optimistic Concurrency - Merging

Two possible scenarios:

1. The version control system is able to merge without help from the user
2. Conflict**:** The version control system needs the user to resolve the conflict

Optimistic Concurrency – SVN merging Options



```
Select: (p) postpone, (df) diff-full, (e) edit,
      (mc) mine-conflict, (tc) theirs-conflict,
      (s) show all options: s

 (e)  edit            - change merged file in an editor
 (df) diff-full       - show all changes made to merged file
 (r)  resolved        - accept merged version of file

 (dc) display-conflict - show all conflicts (ignoring merged version)
 (mc) mine-conflict    - accept my version for all conflicts (same)
 (tc) theirs-conflict  - accept their version for all conflicts (same)

 (mf) mine-full       - accept my version of entire file (even non-conflicts)
 (tf) theirs-full     - accept their version of entire file (same)

 (p)  postpone        - mark the conflict to be resolved later
 (l)  launch          - launch external tool to resolve conflict
 (s)  show all        - show this list
```

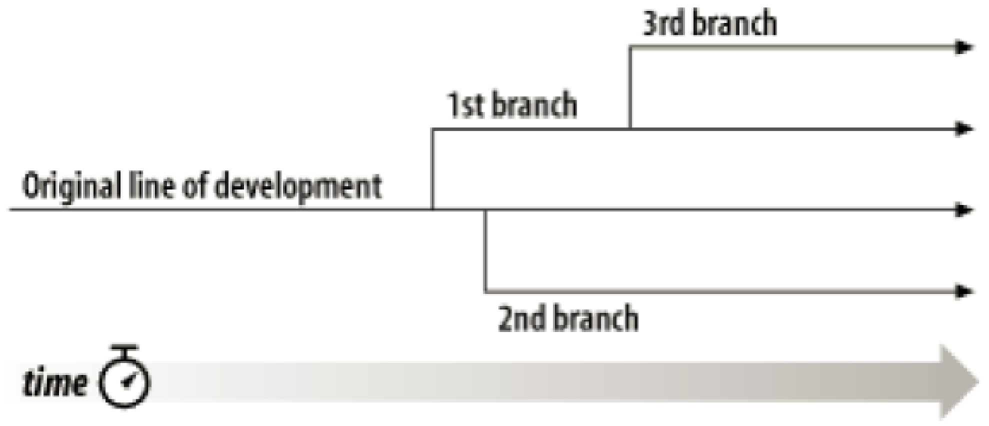
Integrating the Code

What causes merge conflicts?

- Communication issues
- Complex code bases
- Experimental features being built
- Two features being built in the same class by different developers
- Etc.

Version Control – Branching

- Branches are divergent copies of development lines
  - These versions are used to build out complex features, or do experiments, without having an impact on the main code line
- Branching strategies include:
  1. No branching
  2. Release branching
  3. Feature branching



Version Control – Storage Scheme

- Storing every copy of every file we generated over the course of a project is not practical
- Different storage schemes are used. For example:
    - SVN stores incremental differences in files/folder structures
    - Git stores snapshots of the entire project. However, if a file is unchanged, Git uses a link to the previous identical file it has already stored.

Version Control – What's Stored Where

- Server Side: This is out of the scope of this course
- Your local copy contains a special directory
  - It stores (locally) the information needed by the version control system to keep track of your files, version numbers, where the repository is, etc.
  - Needless to say, you should not mess with the contents of this directory.

Version Control – General Rules

1. Update and commit frequently
2. Never break the main branch
3. Always comment clearly what changes are in a revision
4. Test all code before accepting merge
5. Communicate with your team!