CSCB07 - Software Design

Version Control

Based on the slides by Joe Bettridge
Modified by Rawad Abou Assi

Summer 2021

Problems for Developers

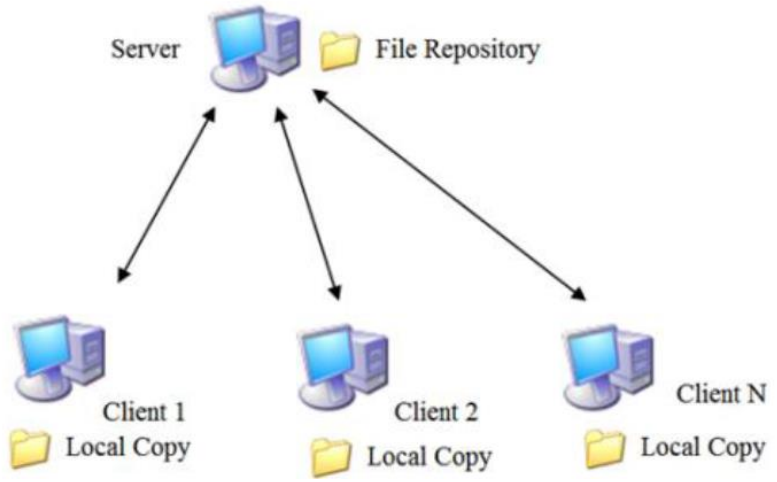
1. How do you keep track of your changes?
   - Don’t keep track
   - Save backups periodically
2. How do you decide who has the authority to make changes?
   - Worry about it after the fact
   - Exchange emails / phone calls
3. How do you keep track of code being worked on at home and in office/lab?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

Solution: Version Control

- Two flavours: Centralized and Distributed
- This course will focus on the centralized flavour

Centralized Version Control

- Keep code in a centralized location (the “Repository”)
- This copy is the “Master Copy”(NEVER directly modify it)
- Create local copies of the repository on each computer you or your team will be working on
- When changes are made to local copy, “Commit” the change to the repository

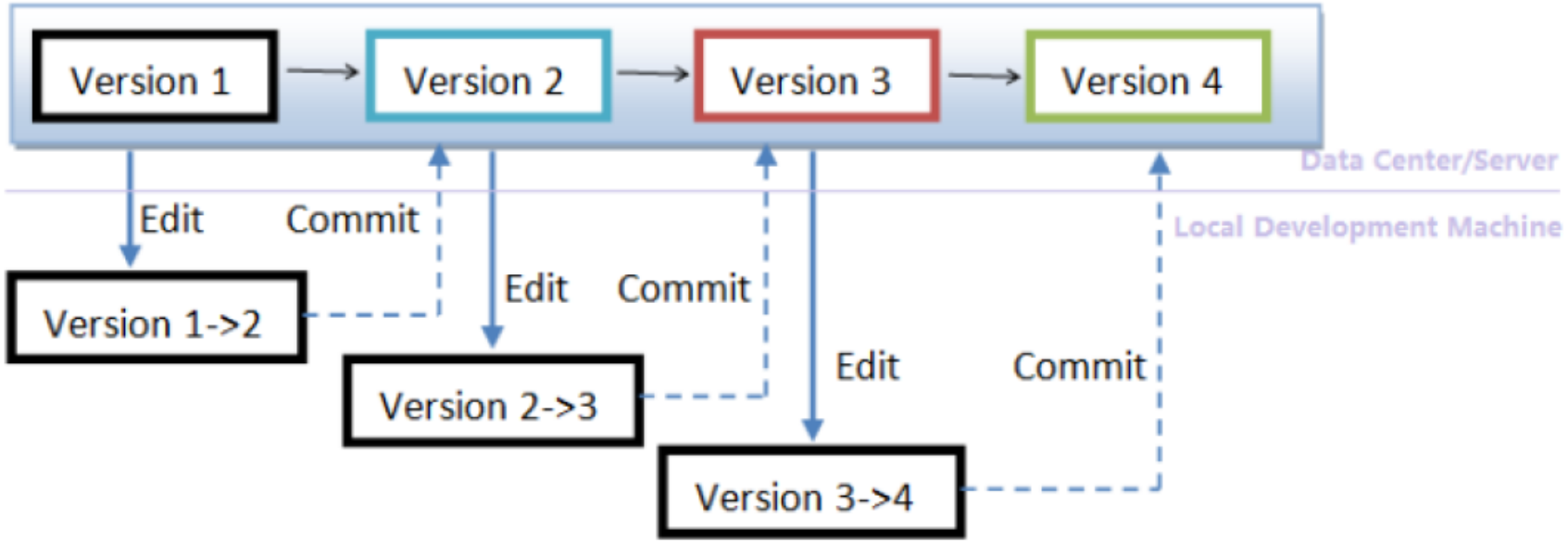


Terminology

- Repository/Repo
- Client program
- Working copy
- Checkout
- Commit

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)



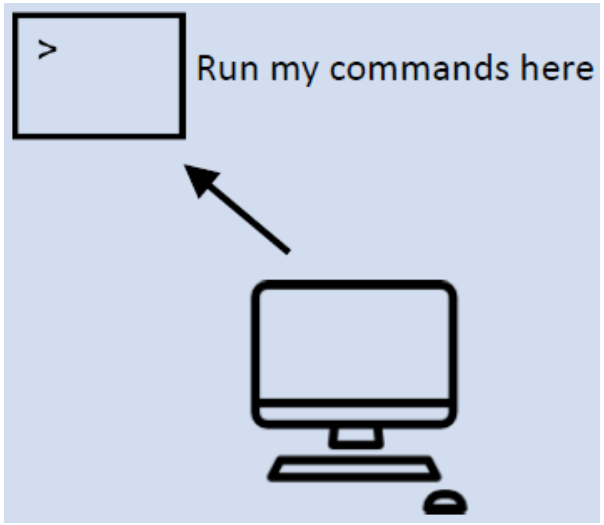
Sub**V**ersion**N** (SVN)

- For this course, we will be using Subversion (SVN)
  - SVN is the successor to Concurrent Versions System (CVS), and was built to help fix many issues in CVS
- Other source control systems include: Git, Mercurial, ClearCase, Perforce, etc.

These are **NOT** version control systems

Secure Shell (SSH)

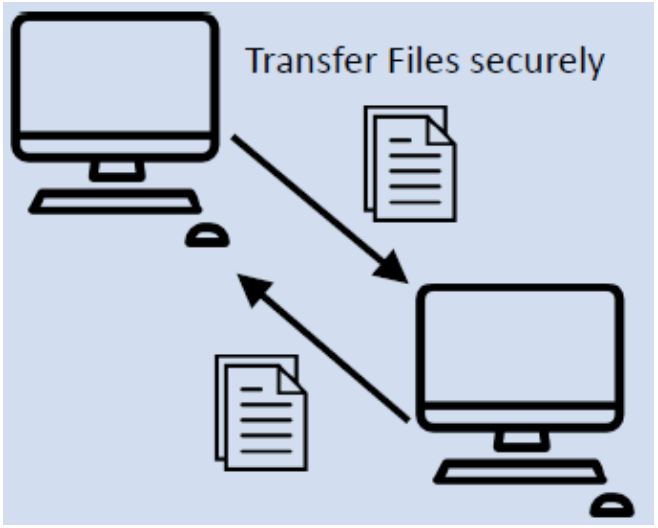
- Connect to remote computer and work in a shell on that computer



Run my commands here

Secure CoPy (SCP)

- Way to securely copy files from one computer to another
- Transfers a copy of the files
- Does not version files

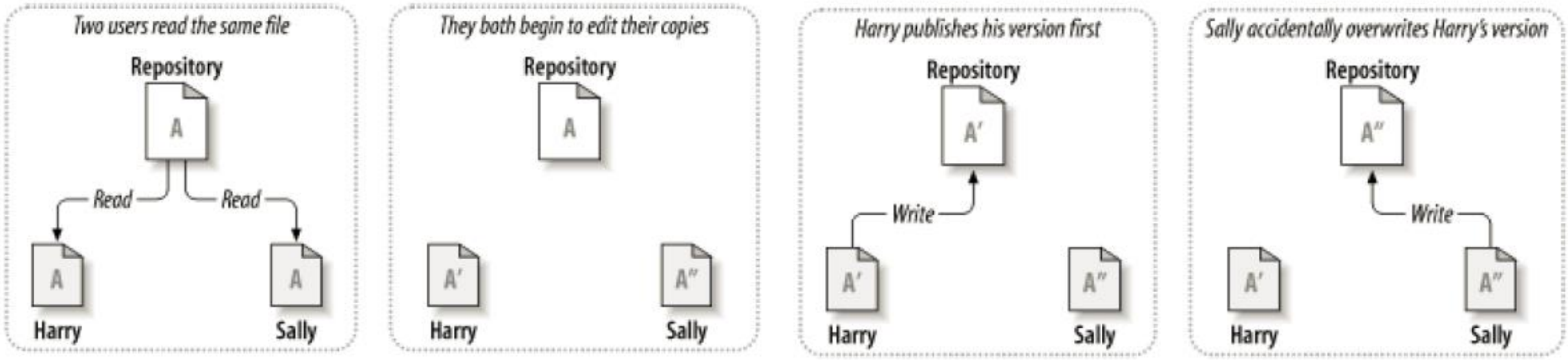


SVN demo

[Demo]

Version Control – Managing Concurrency

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



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

Version Control – Managing Concurrency

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

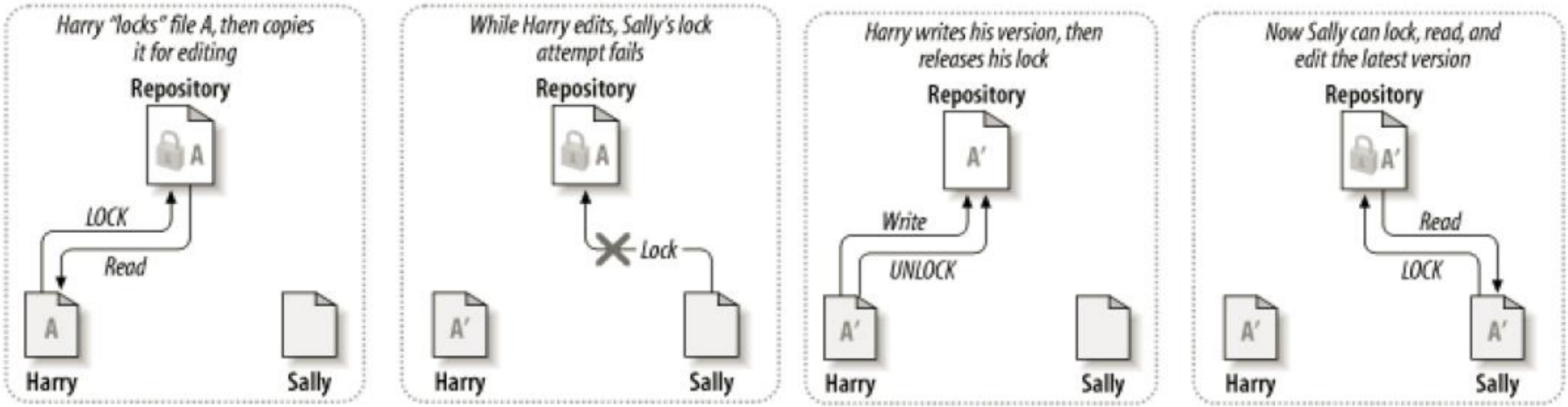
Option 1: Prevent it

- 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

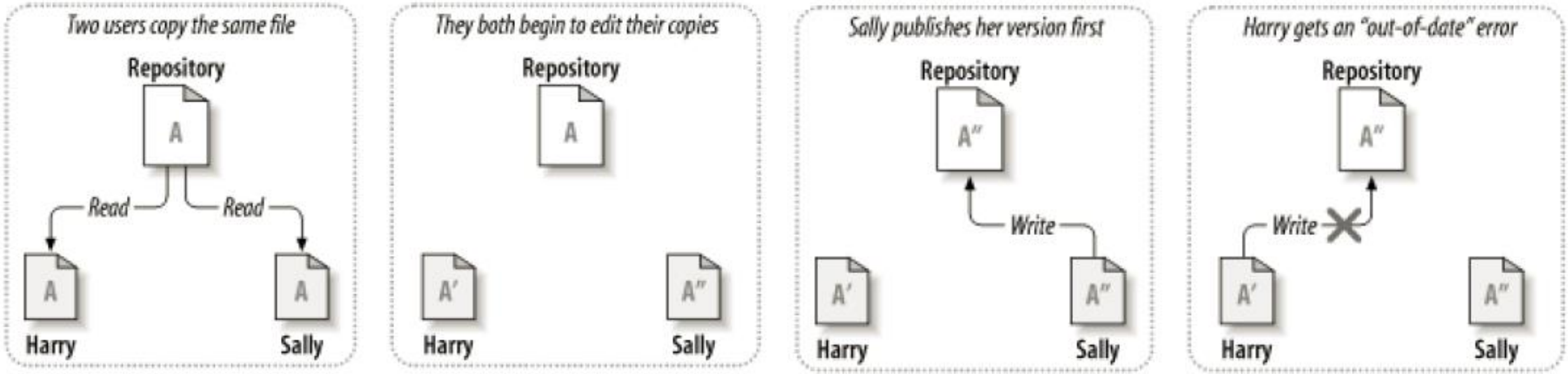
- Optimistic concurrency
- E.g. Subversion, CVS, Perforce

Pessimistic Concurrency



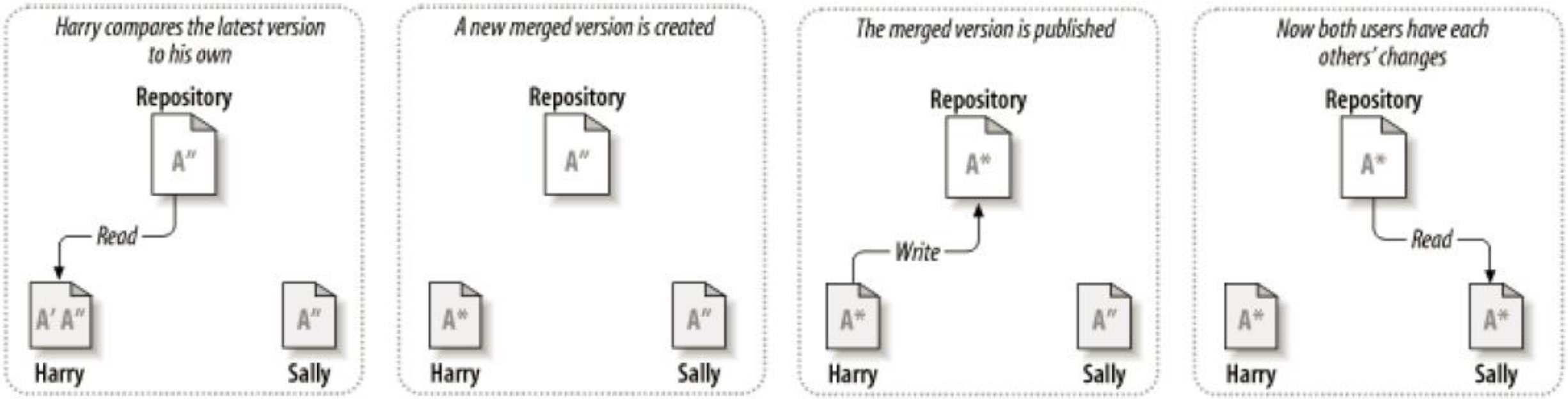
Source: http://svnbook.red-bean.com/

Optimistic Concurrency (1/2)



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

Optimistic Concurrency (2/2)



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

Optimistic Concurrency - Merging

Two possible scenarios:

1. SVN is able to merge without help from the user
2. Conflict**:** SVN needs the user to resolve the conflict

Optimistic Concurrency – 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
```

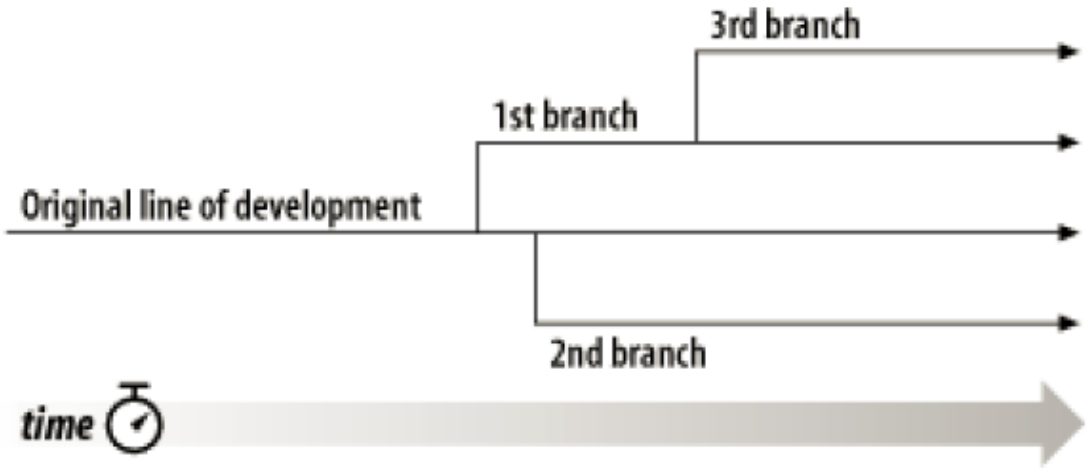
Integrating the Code

What are some reasons merge conflicts happen?

- Communication
- Complex code bases
- Experimental Features being built
- More than one project on the go that impacts this code
- Two features being built in same class by different developers

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
- Version control systems store incremental differences in files/folder structures
  - These differences store enough information to re-construct previous versions, without storing every single copy ever made of the file

Version Control – What's Stored Where

- Server Side: This is out of the scope of this course
- Your local copy contains a special directory **.svn**
  - It stores (locally) the information subversion needs 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. Let subversion do its job

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!