Version Control with Subversion
-- A Tutorial for CS2ME3/SE2AA4

Wen Yu

Feb. 17, 2011

Overview

- Why Version Control?
- Architecture
- Basic Work Cycle
- Submit Your Assignment

Why Version Control?

- Collaboration
- Undoing changes
- Recording history of changes
- Access to files from different computers

Architecture

- Version control system is the management of multiple revisions of the same unit of information
- Version control system is based on typical client-server system
- The repository is the core of a version control system

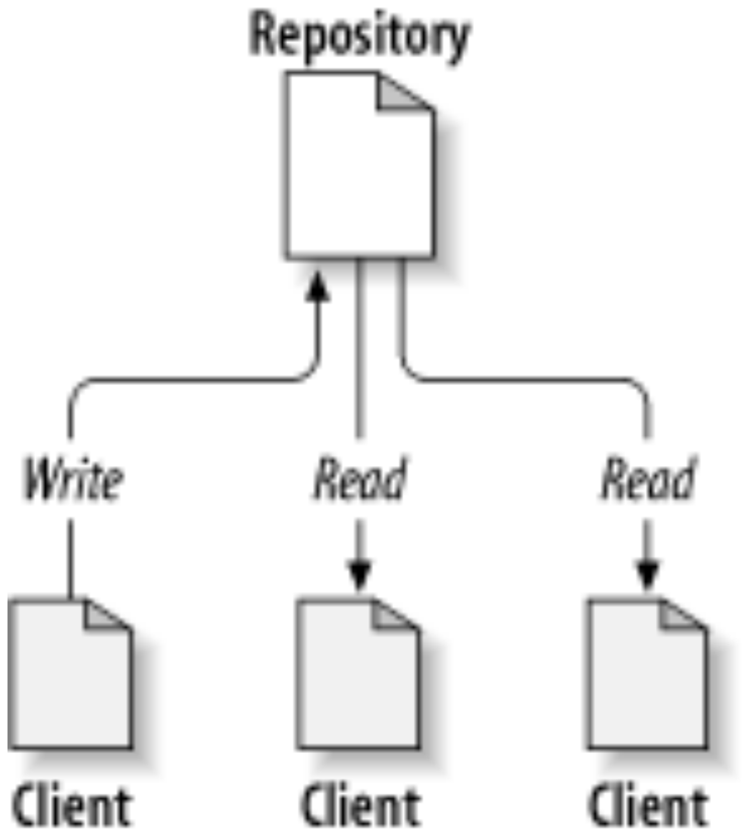
Repository

- Store the most recent copy of files
- Store all the previous revisions of the files
- Log all who make the modifications, and when was the modifications

How Version Control System Works

- Keep the master copy of the file in a central *repository*
- Each author edits a *working copy*
- When they're ready to share their changes, they *commit* them to the repository
- Other people can then do an *update* to get those changes

Typical Client-Sever System



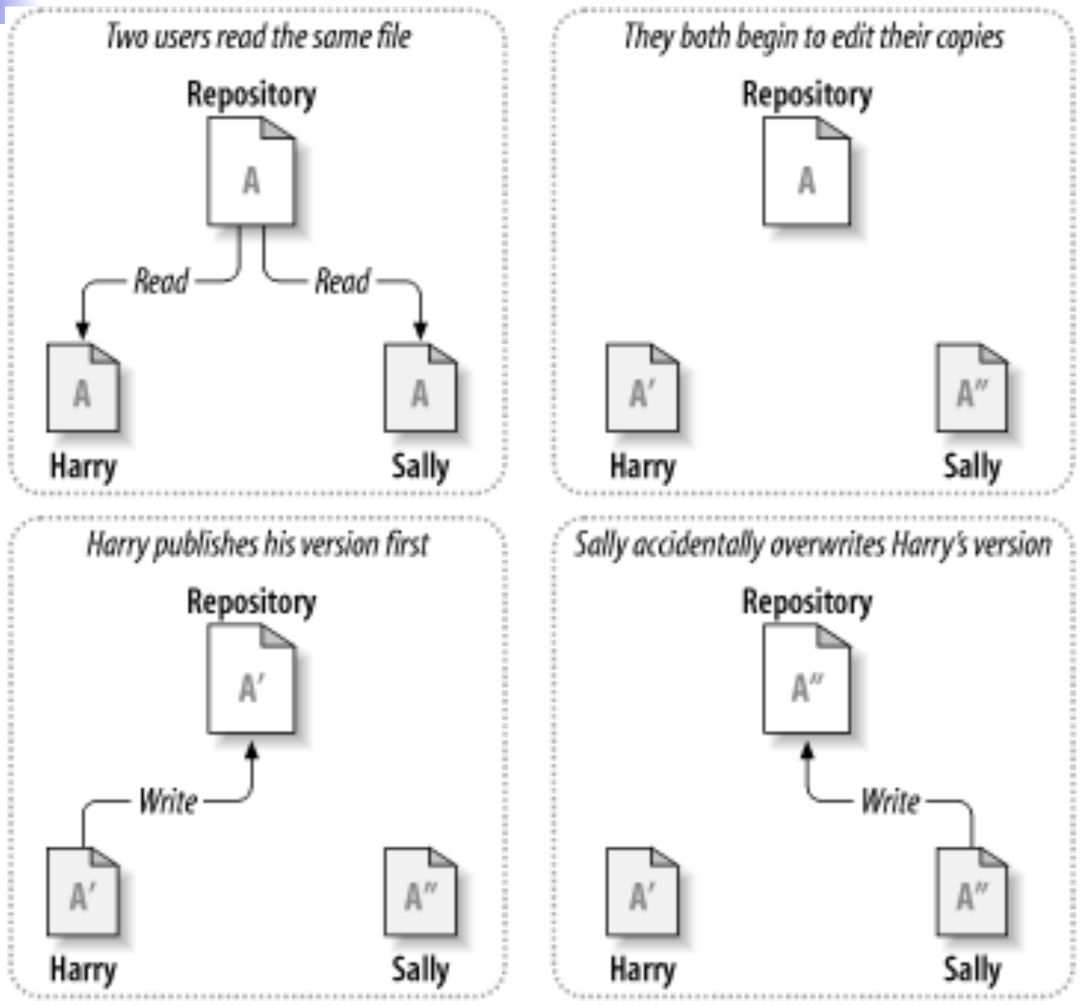
A Typical

Client-server

System

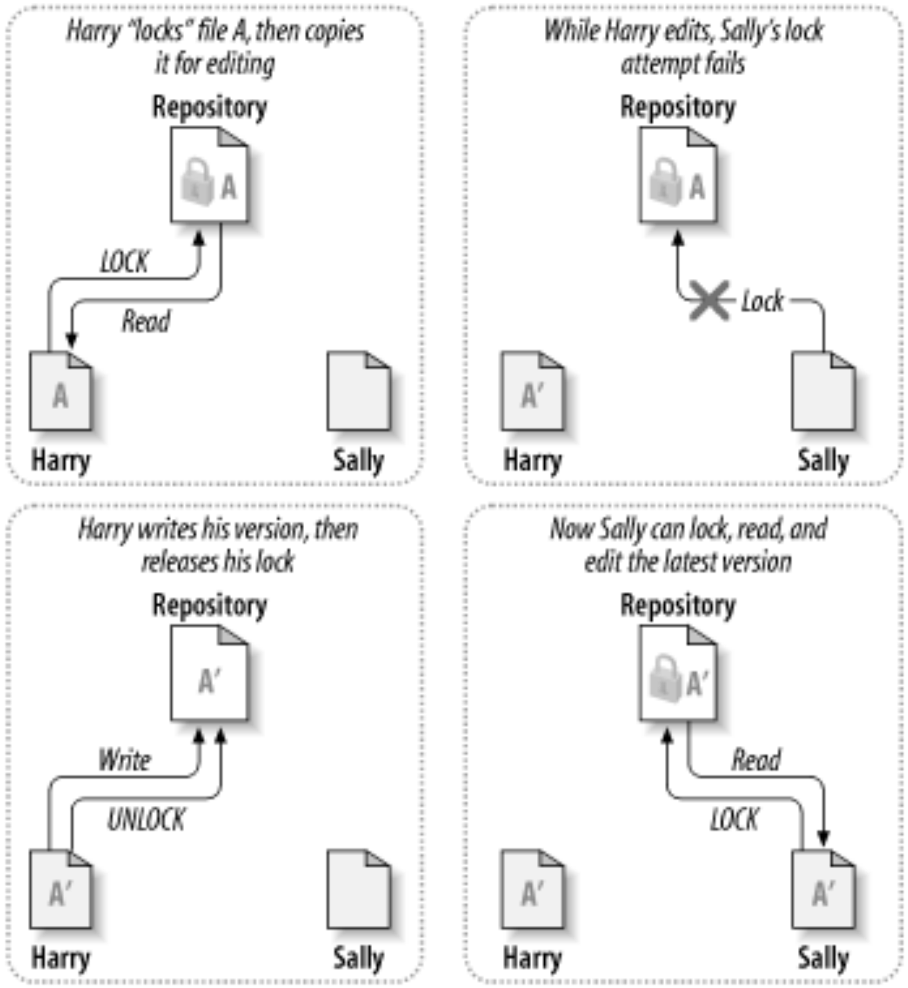
(CSS)

The Problem of File-Sharing


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

1. Two users read the same file
2. They both begin to edit their copies
3. Harry publishes his version first
4. Sally accidentally overwrites Harry's version

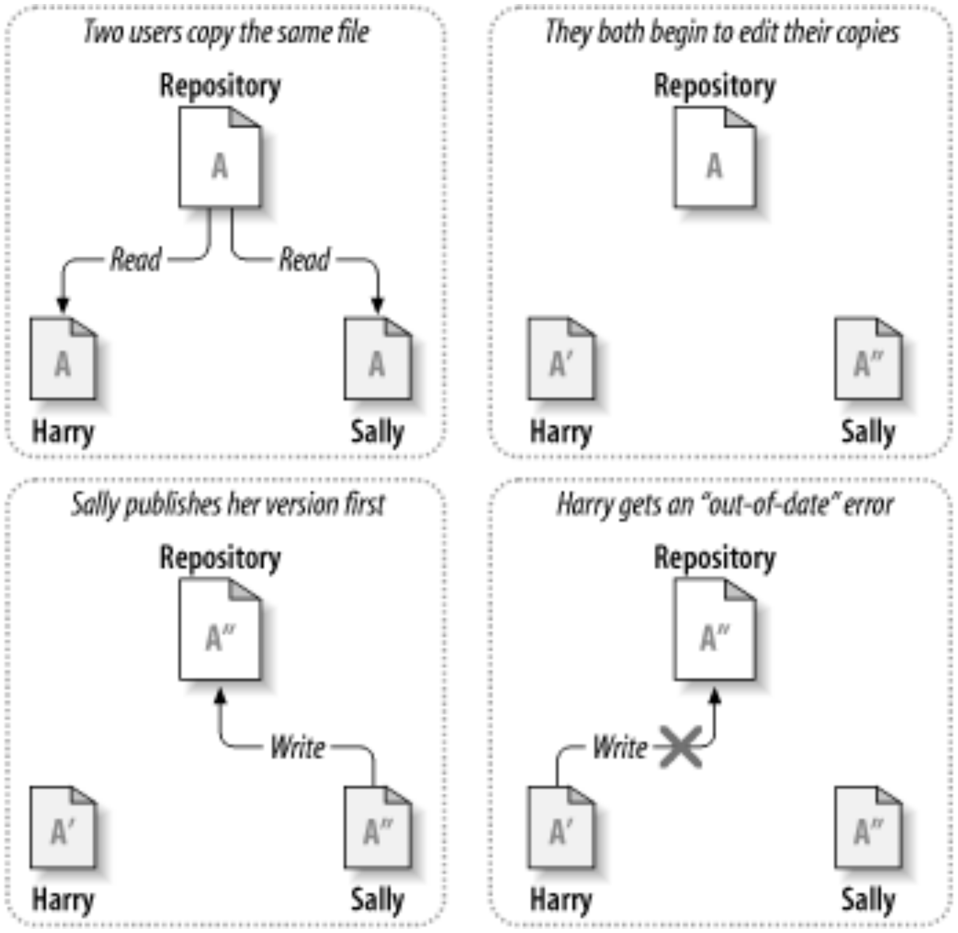
The Lock-Modify-Unlock Solution



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

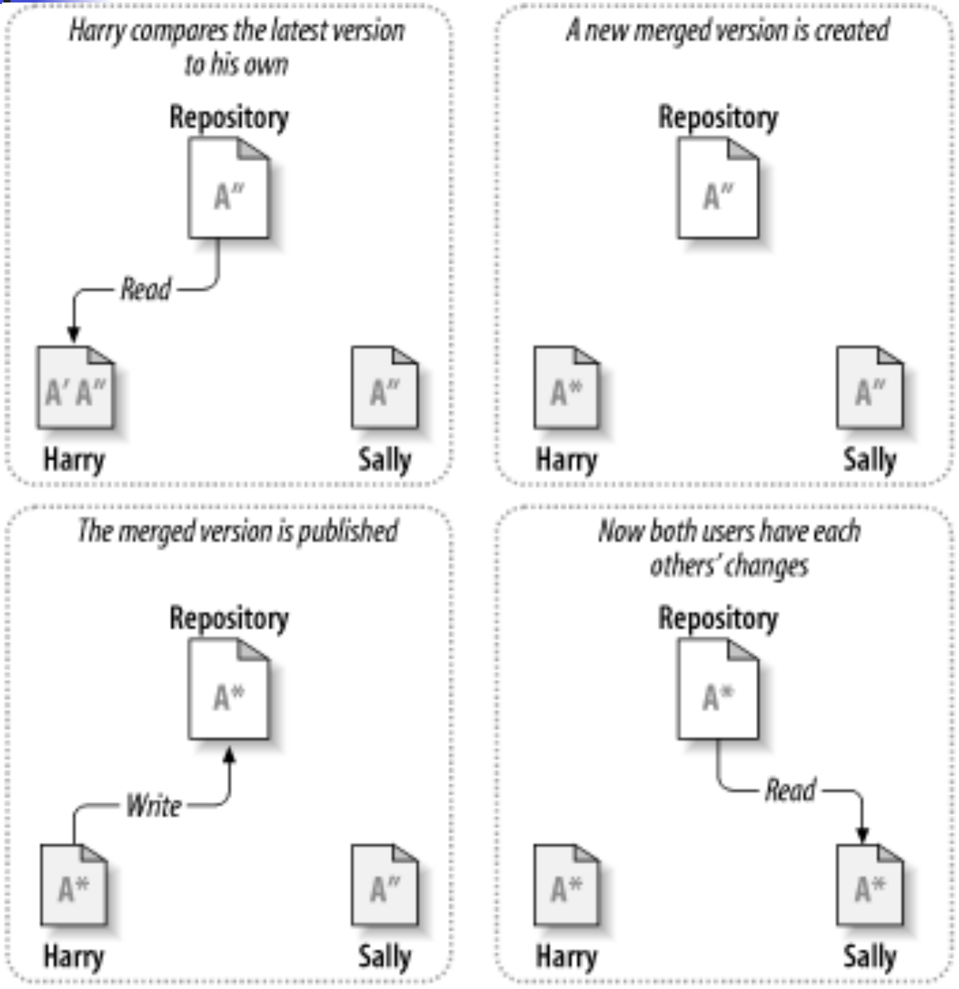
1. Harry "locks" file A, then copies it for editing
2. While Harry edits, Sally's lock attempt fails
3. Harry writes his version, then releases his lock
4. Now Sally can lock, read, and edit the latest version

The Copy-Modify-Merge Solution


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

1. Two users copy the same file
2. They both begin to edit their copies
3. Sally publishes her version first
4. Harry gets an "out-of-date" error

The Copy-Modify-Merge Solution – Cond.


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

5. Harry compares the latest version to his own
6. A new merged version is created
7. The merged version is published
8. Now both users have each other’s change
9. If Sally’s changes overlap with Harry’s changes, Harry may need to discuss with Sally

Basic Work Cycle
1. Update Your Working Copy

- svn update

  Update your working copy to receive any changes made since your last update by other developers on the project

Basic Work Cycle
2. Make Changes

- svn add foo

  Schedule file, directory, or symbolic link *foo* to be added to the repository

- svn delete foo

  Schedule file, directory, or symbolic link *foo* to be deleted from the repository

Basic Work Cycle
2. Make Changes – Cond.

- svn copy foo bar

  Create a new item *bar* as a duplicate of *foo*

- svn move foo bar

  Schedule *bar* for addition as a copy of *foo*, and schedule *foo* for removal

Basic Work Cycle
3. Exam Your Change

- svn status

  Detect all file and tree changes you've made

- svn diff

  Print out file changes in unified diff format

Basic Work Cycle
4. Merge Others' Changes

- svn update

  Check if changes from the server overlapped with your own
- Whenever a conflict occurs you have to merge conflicts by hand
- svn resolved

  Let Subversion know you've resolved the conflict

Basic Work Cycle
5. Possibly Undo Some Changes

- svn revert

  Undo *any* scheduled operations

Basic Work Cycle
6. Commit Your Changes

- svn commit –m “your commit”

  -- The command sends all of your changes to the repository

  -- When you commit a change, you need to supply a *log message*, describing your change

Examining History

- svn log

  Shows log messages, and which paths changed
- svn diff
- svn cat

  Retrieve and display any file as it existed in a particular revision number
- svn list

  Displays the files in a directory for any given revision.

Submit Your Assignment: 1. Checkout the Repository

- Change to the directory that you would like to work with
- Check out your repository

  svn checkout https://websvn.mcmaster.ca/cs2me3-se2aa4/ <your macid>

  -- Create a new "working copy" of the data, a sort of private workspace

```
[yuw4@moore ~/TA] svn co https://websvn.mcmaster.ca/cs2me3-se2aa4/yuw4
Checked out revision 41.
[yuw4@moore ~/TA] ls
yuw4/
```

Submit Your Assignment: 2. Add a Folder

- Change to your working directory
- Add a folder for your assignment 1, such as a1
- Commit the new folder to your repository

```
[yuw4@moore ~/TA] cd yuw4/
[yuw4@moore yuw4] mkdir a1
[yuw4@moore yuw4] ls
a1/
[yuw4@moore yuw4] svn add a1
A         a1
[yuw4@moore yuw4] svn ci -m "Add a folder a1 for assignment 1."
Adding         a1
Committed revision 42.
```

Submit Your Assignment: 3. Add Files

- Change to the new directory
- Create and add the files, such as st.py
- Commit the file to your repository

```
[yuw4@moore yuw4] cd a1
[yuw4@moore a1] ls
st.py
[yuw4@moore a1] svn add st.py
A         st.py
[yuw4@moore a1] svn ci -m "Add a python file for assignment 1."
Adding         a1/st.py
Transmitting file data .
Committed revision 43.
[yuw4@moore a1] svn cat st.py
# CS2ME3 Assignment 1
```

Submit Your Assignment: 4. Make Changes

- Modify your work
- Update your repository
- Commit your work

```
[yuw4@moore a1] cat st.py
# CS2ME3 Assignment 1
# Name: Wen Yu
[yuw4@moore a1] svn cat st.py
# CS2ME3 Assignment 1
[yuw4@moore a1] svn ci -m "The file st.py is modified."
Sending        a1/st.py
Transmitting file data .
Committed revision 44.
[yuw4@moore a1] svn cat st.py
# CS2ME3 Assignment 1
# Name: Wen Yu
```

Good to Know

- You need a subversion client
- You only need to check out your repository to your working space once
- Get the habit of committing your work frequently
- Get help

```
svn help
```

List all the subcommands

```
svn help <subcommand>
```

Describe syntax, switches, and behavior of subcommand

References

- Ben Collins-Sussman, Brain W. Fitzpatrick, C. Micheal Pilatoversion, Control with Subversion,

  http://svnbook.red-bean.com
- Python Software Foundation, Software Carpentry, http://www.swc.scipy.org