The software versioning and revision control system *subversion*[1]

Matthias Sommer

Hans Ertel Centre for Weather Research
Data Assimilation Branch
LMU München

March 14, 2013

Hans-Ertel-Zentrum für Wetterforschung
Deutscher Wetterdienst

[1] Much of this content is from `http://svnbook.red-bean.com`

Everything on one slide!

Subversion...

- ... is a free, cost-free and open-source software versioning and revision control system.
- ... administers code, files, folders, i. e. it saves different versions.
- ... is a successor of *CVS*.
- ... works with a central repository (unlike *Mercurial* or *Git*).
- ... and local clients.

It is for anyone who ...

- ... wishes to keep track of different version of his/her work (=*files or folders*).
- ... wishes to share work between different users.
- ... has access to a central repository (can be local).

Pros and cons

+ Reproducible work
+ Easy sharing even with very large projects
+ No more `newcode_old_tmp_uralt_2.405.f90`
- One more piece of software.

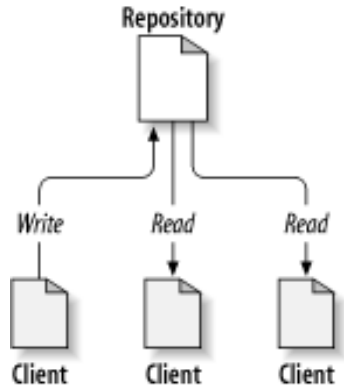
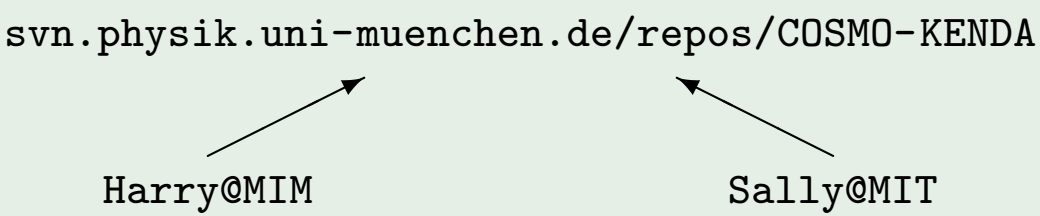


Figure: Like a server but all versions are saved

Example

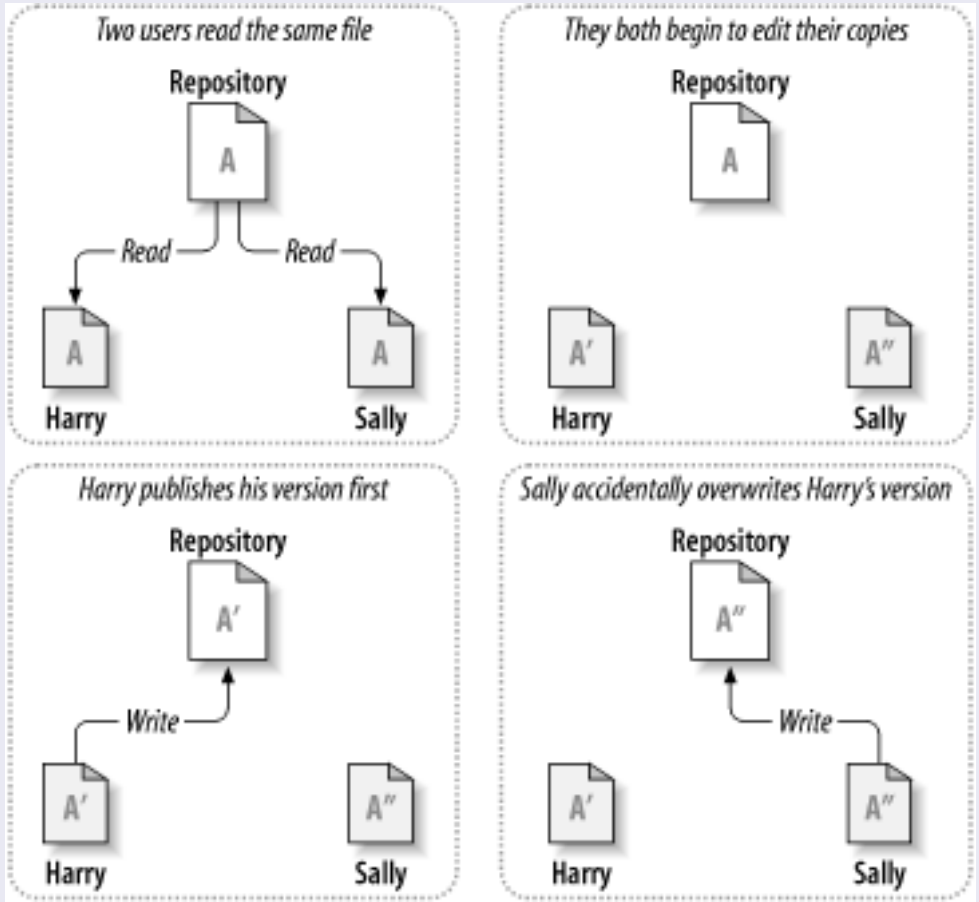
Example (Harry and Sally are both working on some matlab scripts)

- They have a central repository on the physics server...
- ...and both their local work copies on their computers.
- To manage the code, they use *subversion* on the command line or with the GUI *kdesvn*.

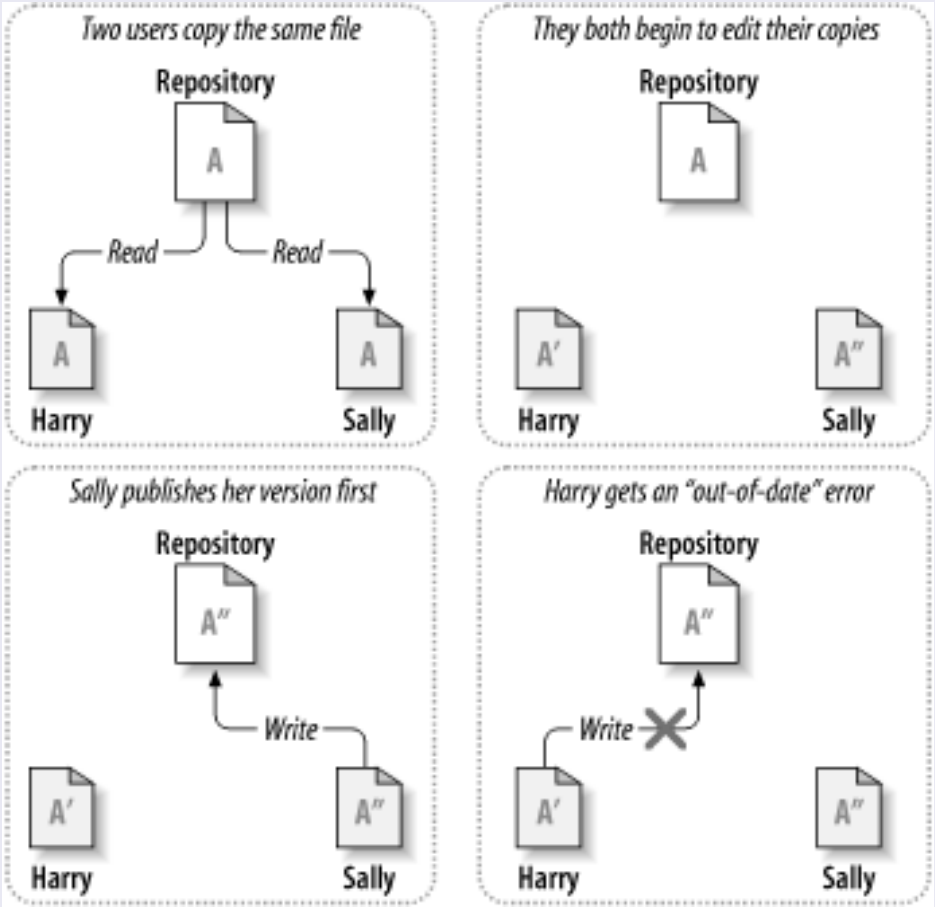


- No need to send code via e-mail or putting it on a public folder.
- No need to tell each other that we have done some changes, neither which nor why.
- Going back to an earlier (and working) version is easy even with dependencies.

Two users working simultaneously on a file

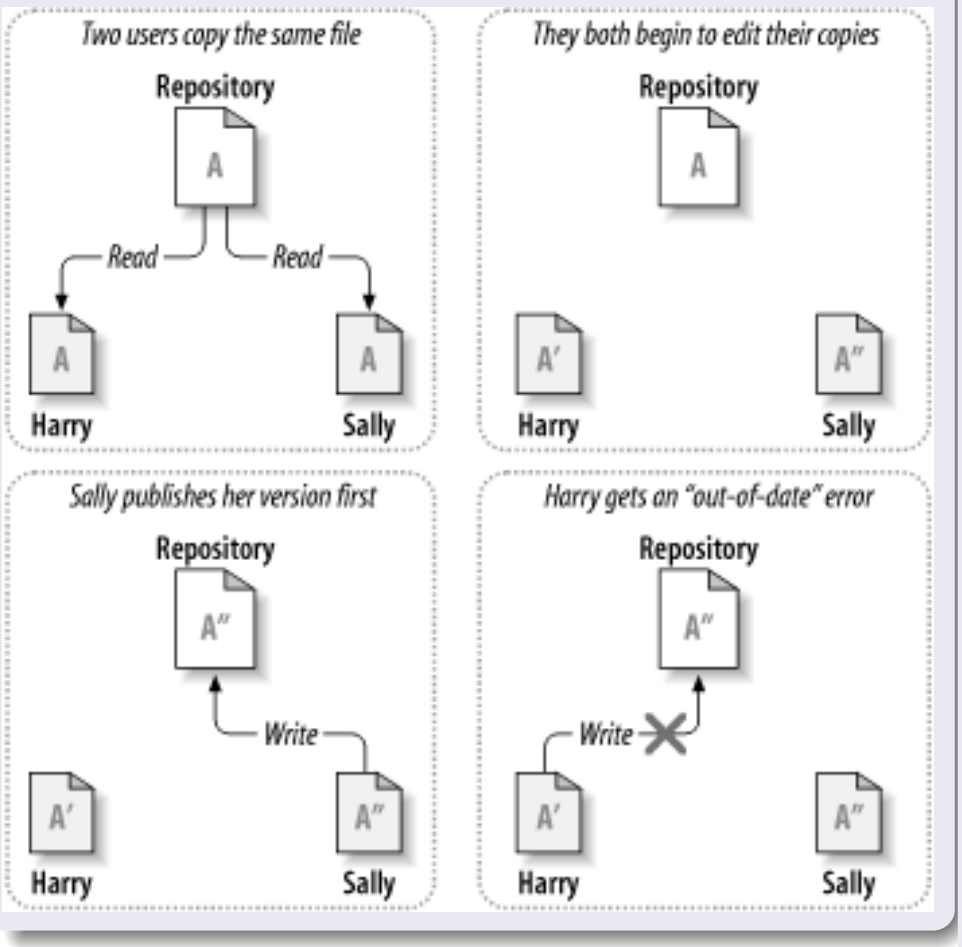


A conflict

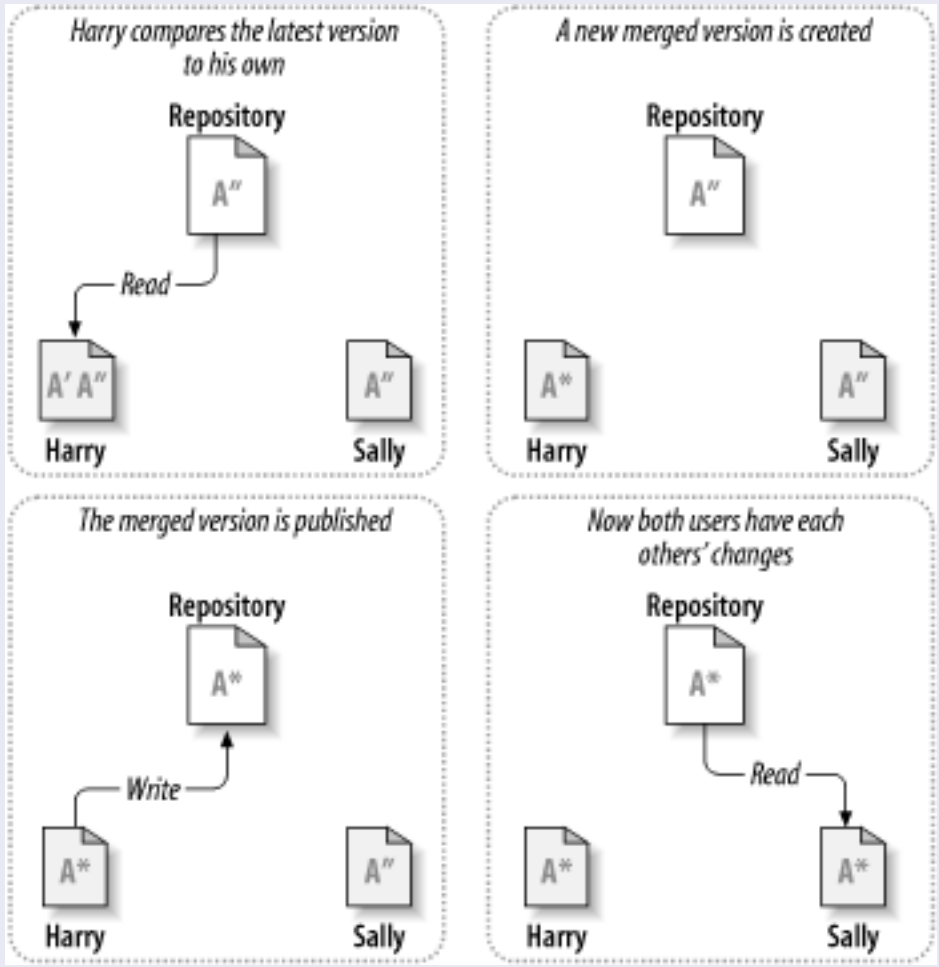

Two users copy the same file
Repository
A
Read
Read
A
A
Harry
Sally
They both begin to edit their copies
Repository
A
A′
A″
Harry
Sally
Sally publishes her version first
Repository
A″
Write
A′
A″
Harry
Sally
Harry gets an "out-of-date" error
Repository
A″
Write
A′
A″
Harry
Sally

The solution

A conflict



Resolving the conflict



Typical work steps

Initially (technically equivalent to later actions)

- Generate repository
- Upload directory with ('put under version control', 'initial check in')
- Download directory ('initial check out')

Everyday work

- Work on the files (e. g. add or change code) with arbitrary software (e. g. any text editor).
- Update work copy ('check out'): Downloads the head (most recent version) of all files
  - Files in local work copy that are identical to head: Nothing happens
  - Files in local work copy that have been changed only by others: Download head
  - Files in local work copy that have been changed by others and locally: Resolve conflict automatically or manually
- Commit changes ('check in')
  - Files that have been changed locally: Upload

Special tasks

- Check status
- Reset to earlier version
- Compare to earlier version

The repository ('Projektarchiv')

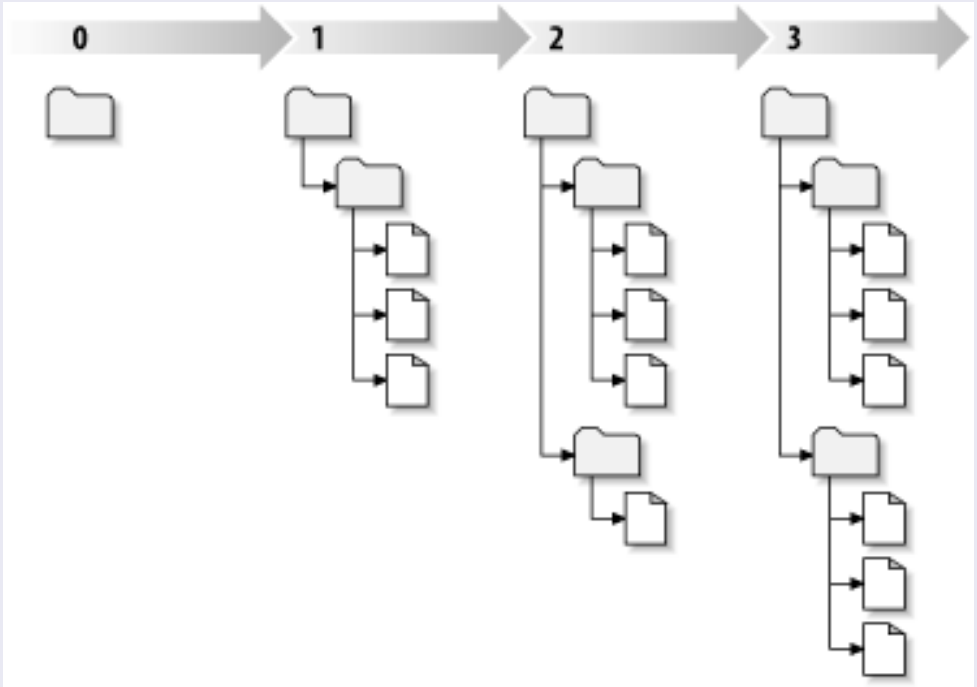


Figure: The repository ('Projektarchiv')

Online demonstration: Basics

Example (Initial checkout and simple transfer)

User 1	User 2
•	• Initial Check out
• Modify hello_world.f90	•
• Check in	•
•	• Check out: No local changes → No conflict

Online demonstration: Conflict resolution

Example (Automatic conflict resolution)

User 1

- Modify Line 2 of hello_world.f90
- Check in
-
-
- Check out: No local changes → No conflict

User 2

- Modify Line 4 of hello_world.f90
-
- Check out: Automatic conflict resolution
- Check in
-

Example (Manual conflict resolution)

User 1

- Modify Line 4 of hello_world.f90
- Check in
-
-
- Check out: No local changes → No conflict

User 2

- Modify Line 4 of hello_world.f90
-
- Check out: Manual conflict resolution with kdiff3
- Check in
-

Trunk, branches and tags

Trunk

- Main development line

Branches

- A ‘side line’ of a project
- Interchange some changes with trunk but not all of them
- Technically identical to trunk

Tags

- Possibility for giving static names to certain version numbers
- E. g. `my_project_2.35` = Version 109 of `my_project`

Summary

Remember

- Update your local working copy to head ('Download'): Check out
- Commit your local changes to repository ('Upload'): Check in
- No file on the working copy or the repository is changed automatically
- While the syntactic consistency is ensured, the semantic is not

Example (svn-Repositories)

- `svn.physik.uni-muenchen.de/repos/COSMO-KENDA`
- `svn.physik.uni-muenchen.de/repos/COSMO`
- `svn.zmaw.de/svn/osas/trunk/3dvar`

Credits