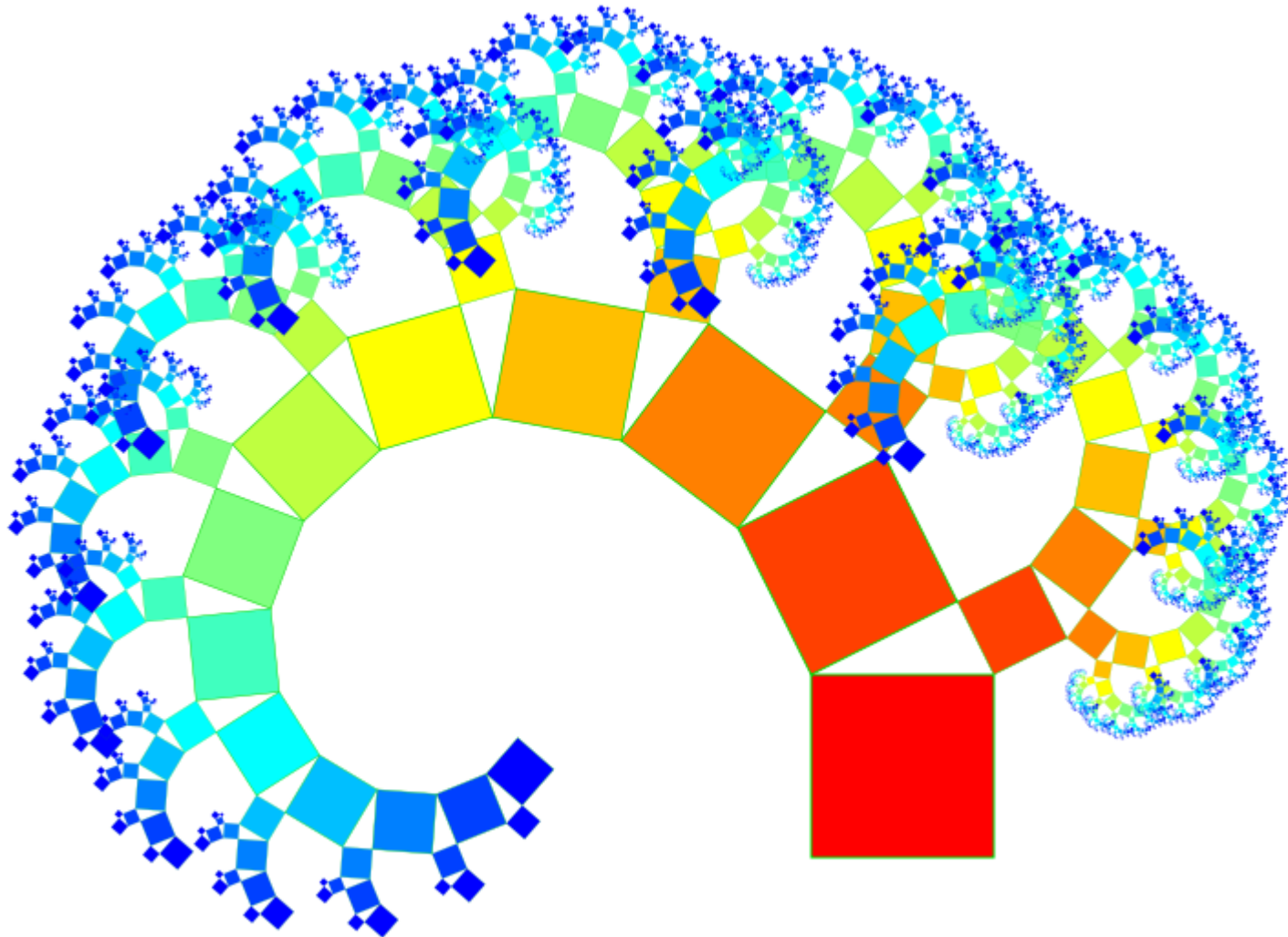


Version Control



2011 May 26 – talk starts at 3:10



The speaker

Davide Del Vento, PhD in Physics
Consulting Services, Software Engineer
NCAR - CISL

<http://www2.cisl.ucar.edu/uss/csg>

office: Mesa Lab, Room 42B

phone: (303) 497-1233

email: ddvento@ucar.edu

ExtraView Tickets: cislhelp@ucar.edu

Cover Picture: http://commons.wikimedia.org/wiki/File:Pythagoras_tree_1_2_12_jet.svg

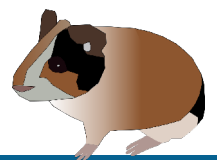
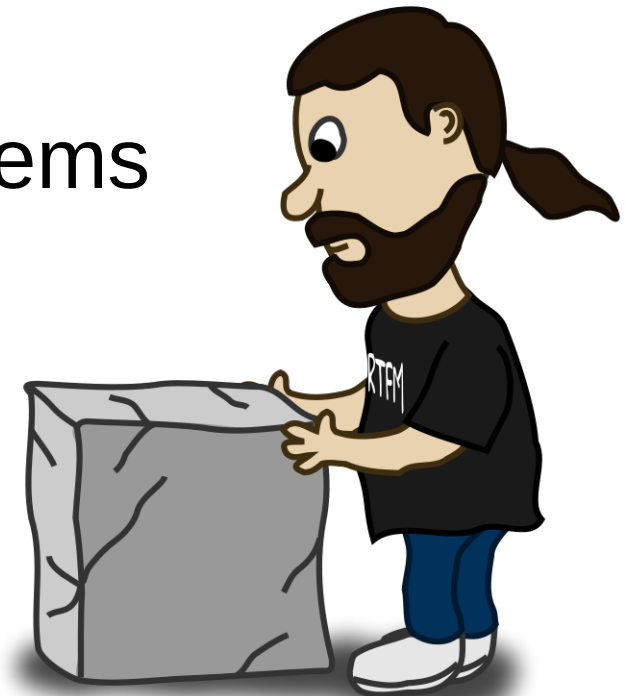


Table of Contents

- Why version control?
- Centralized version control systems
 - Manually copying files
 - SVN (subversion)
- Distributed version control systems
 - HG (mercurial)
 - BZR (bazaar)
 - GIT
- References



Why doing version control?

- User: “This worked last month, now it doesn't work anymore. HELP!”
- Me: “What did you change?”
- User: “I don't remember” (or “Nothing” or “It doesn't matter” or “It's your supercomputer to have something wrong”)

Why doing version control?

- User: “This worked last month, now it doesn't work anymore. HELP!”
- Me: “What did you change?”
- User: “I don't remember” (or “Nothing” or “It doesn't matter” or “It's your supercomputer to have something wrong”)

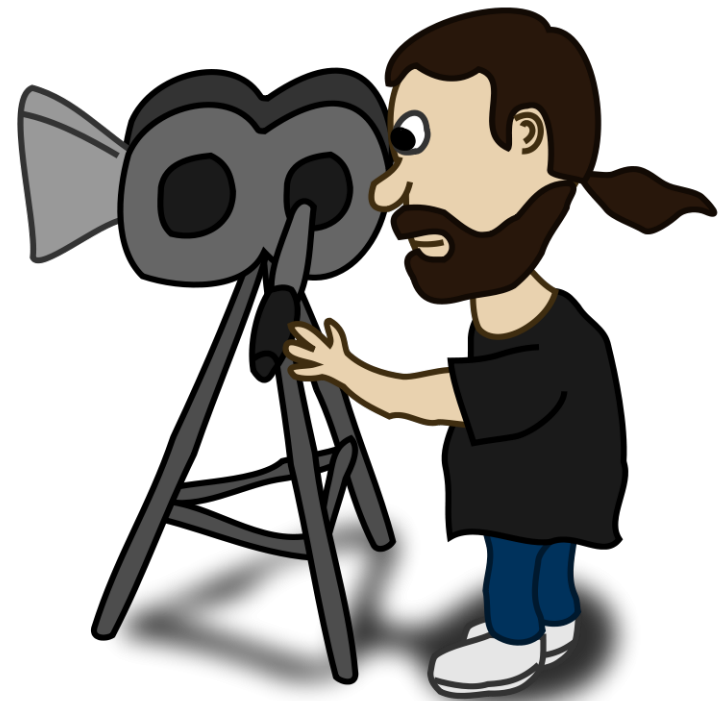


Why doing version control?

- User: “This worked last month, now it doesn't work anymore. HELP!”
- Me: “What did you change?”
- User: “I don't remember” (or “Nothing” or “It doesn't matter” or “It's your supercomputer to have something wrong”)
- 99% - it's later discovered the user changed something that broke the code

Why doing version control?

- To keep track of what has changed!



Why doing version control?

- To keep track of what has changed!
- Side benefits:
 - release management
 - better communications
 - backup of the entire history
 - concentrate on the project not on the risk of breaking it (i.e. more open to experiment)

Why doing version control?

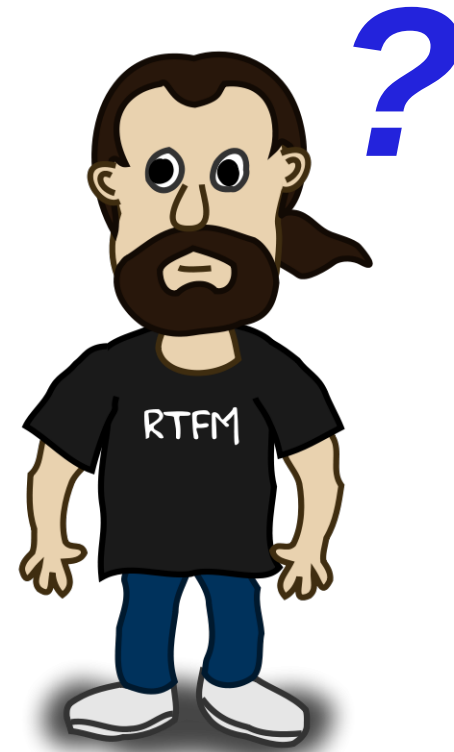
- To keep track of what has changed!
- Side benefits:
 - release management
 - better communications
 - backup of the entire history
 - concentrate on the project not on the risk of breaking it (i.e. more open to experiment)
- How?

Manually copying files

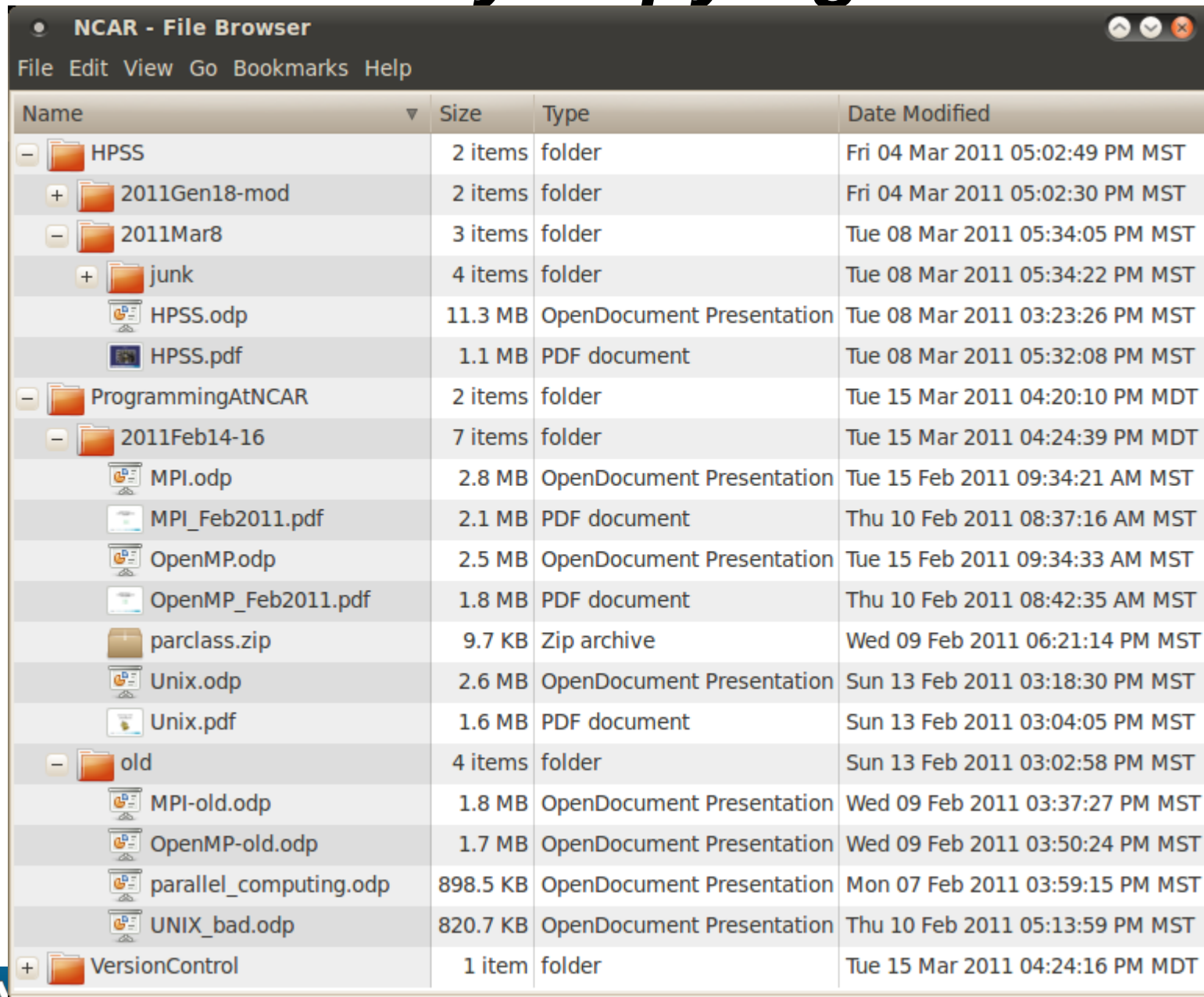
- A possible solution to this problem:
- Just make a copy of the files/directories

Manually copying files

- A possible solution to this problem:
- Just make a copy of the files/directories
- But when?
 - at any “relevant” time,
 - or at regular intervals,
 - when you remember...



Manually copying files



The screenshot shows a web-based file browser interface for NCAR. The window title is "NCAR - File Browser". The menu bar includes "File", "Edit", "View", "Go", "Bookmarks", and "Help". The main content area displays a table of files and folders with columns for Name, Size, Type, and Date Modified. The files are organized into a hierarchical structure with expandable/collapsible folders indicated by minus/plus icons.

Name	Size	Type	Date Modified
[-] HPSS	2 items	folder	Fri 04 Mar 2011 05:02:49 PM MST
[+] 2011Gen18-mod	2 items	folder	Fri 04 Mar 2011 05:02:30 PM MST
[-] 2011Mar8	3 items	folder	Tue 08 Mar 2011 05:34:05 PM MST
[+] junk	4 items	folder	Tue 08 Mar 2011 05:34:22 PM MST
HPSS.odp	11.3 MB	OpenDocument Presentation	Tue 08 Mar 2011 03:23:26 PM MST
HPSS.pdf	1.1 MB	PDF document	Tue 08 Mar 2011 05:32:08 PM MST
[-] ProgrammingAtNCAR	2 items	folder	Tue 15 Mar 2011 04:20:10 PM MDT
[-] 2011Feb14-16	7 items	folder	Tue 15 Mar 2011 04:24:39 PM MDT
MPI.odp	2.8 MB	OpenDocument Presentation	Tue 15 Feb 2011 09:34:21 AM MST
MPI_Feb2011.pdf	2.1 MB	PDF document	Thu 10 Feb 2011 08:37:16 AM MST
OpenMP.odp	2.5 MB	OpenDocument Presentation	Tue 15 Feb 2011 09:34:33 AM MST
OpenMP_Feb2011.pdf	1.8 MB	PDF document	Thu 10 Feb 2011 08:42:35 AM MST
parclass.zip	9.7 KB	Zip archive	Wed 09 Feb 2011 06:21:14 PM MST
Unix.odp	2.6 MB	OpenDocument Presentation	Sun 13 Feb 2011 03:18:30 PM MST
Unix.pdf	1.6 MB	PDF document	Sun 13 Feb 2011 03:04:05 PM MST
[-] old	4 items	folder	Sun 13 Feb 2011 03:02:58 PM MST
MPI-old.odp	1.8 MB	OpenDocument Presentation	Wed 09 Feb 2011 03:37:27 PM MST
OpenMP-old.odp	1.7 MB	OpenDocument Presentation	Wed 09 Feb 2011 03:50:24 PM MST
parallel_computing.odp	898.5 KB	OpenDocument Presentation	Mon 07 Feb 2011 03:59:15 PM MST
UNIX_bad.odp	820.7 KB	OpenDocument Presentation	Thu 10 Feb 2011 05:13:59 PM MST
[+] VersionControl	1 item	folder	Tue 15 Mar 2011 04:24:16 PM MDT

Manually copying files

- Actually not as bad as it may sound
- You can work this way if you are really organized and tidy
- In fact at least two large open-source projects successfully work(ed) this way:
 - The linux kernel (now they switched to git)
 - The bash shell

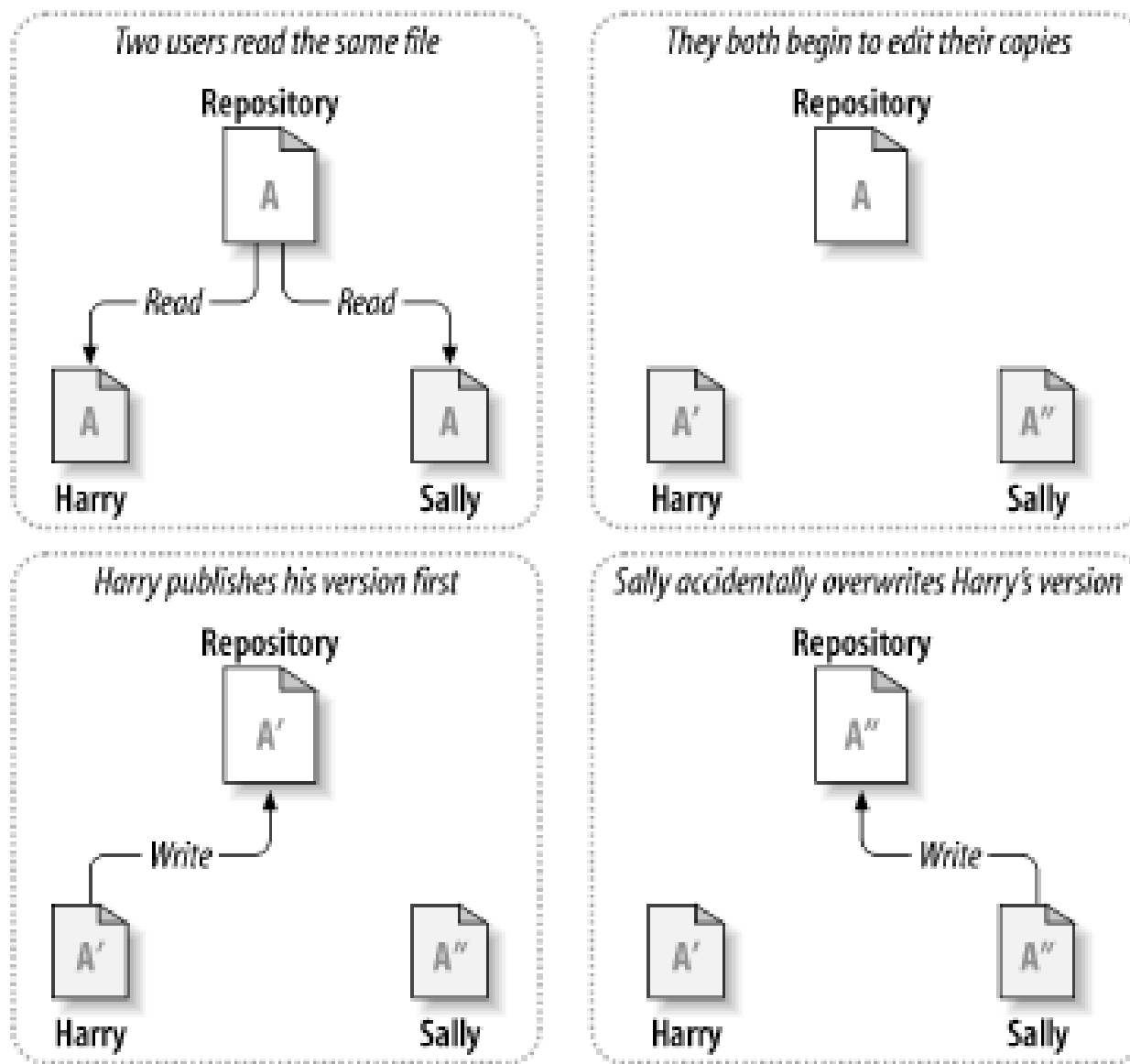
Manually copying files

- Actually not as bad as it may sound
- You can work this way if you are really organized and tidy
- In fact at least two large open-source projects successfully work(ed) this way:
 - The linux kernel (now they switched to git)
 - The bash shell
- But I'm not going to teach you this

Typical workflow (manual)

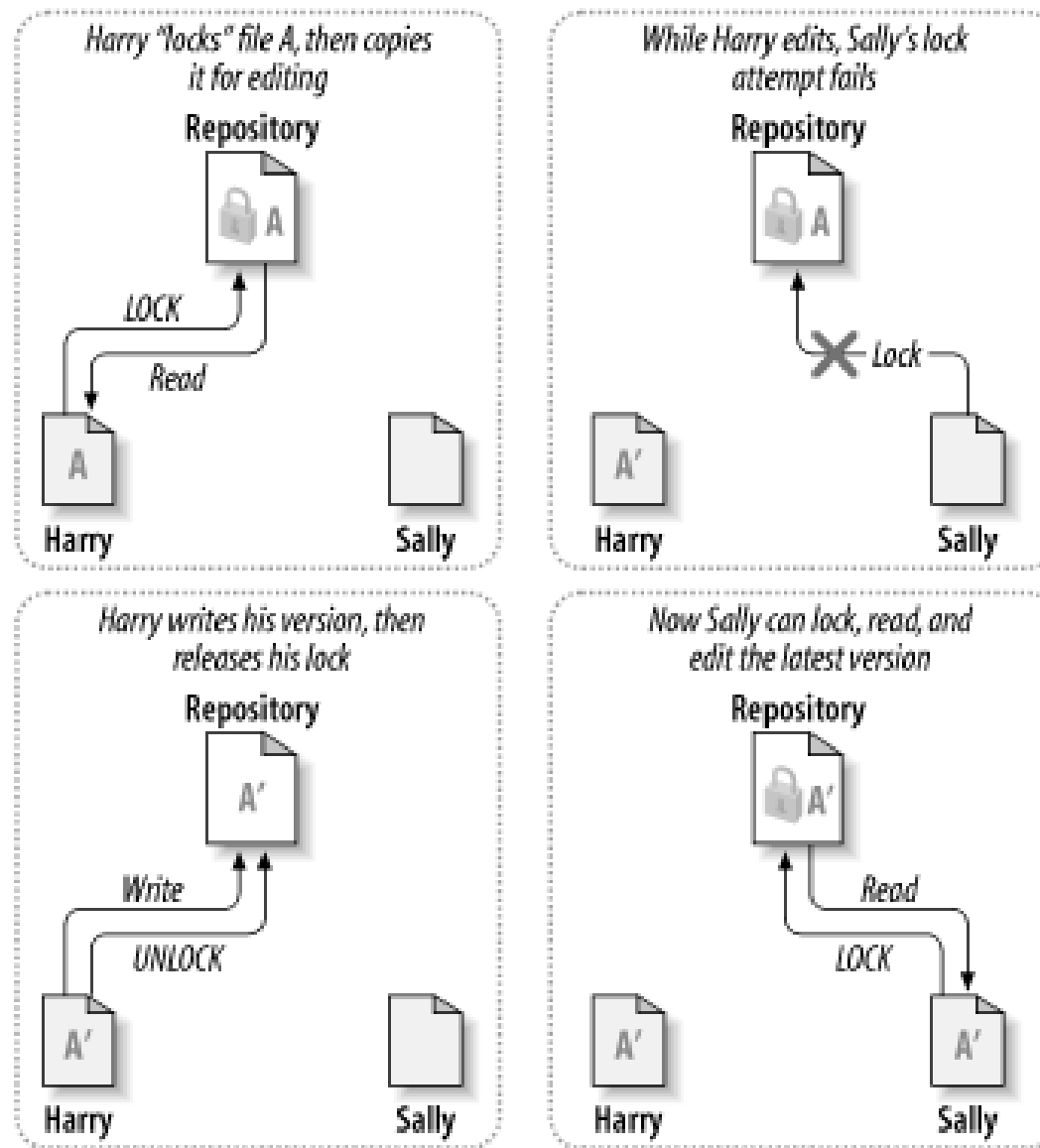
- 1) Get the latest version
- 2) “Lock” the whole repository
 - ideally just the files you're going to change
- 3) Change your files
- 4) If not happy, consider reverting changes
- 5) If not happy, goto 3)
- 6) “Unlock” and post the new stuff

Lock-Modify-Unlock workflow (1)



source: subversion “red” book

Lock-Modify-Unlock workflow (2)



source: subversion "red" book

Manually copying files pros/cons

- ✓ You don't need to learn yet another tool
- ✓ You'll have full control on what you do
- ✓ You may use it as a backup tool
- ✓ You can write your own scripts
- ✓ It's much better than nothing

Manually copying files pros/cons

- ✓ You don't need to learn yet another tool
- ✓ You'll have full control on what you do
- ✓ You may use it as a backup tool
- ✓ You can write your own scripts
- ✓ It's much better than nothing
- ✗ You have to be organized (e.g. naming conventions)
- ✗ You'll end up with a lot of copies, wasting disk space
- ✗ Sifting through the copies looking for something is hard
- ✗ Poor granularity
- ✗ No metadata

Manually copying files pros/cons

- ✓ You don't learn yet tool
- ✗ All the pros are lies:
- ✗ What if you need to find when you changed a particular line in a particular file?
- ✗ What if you need more granularity?
- ✗ What if you work on two features at the same time? How do you manage the merge?
- ✗ What if you have conflicts?
- ✗ Even just showing a diff is a pain

Centralized version control



Centralized version control

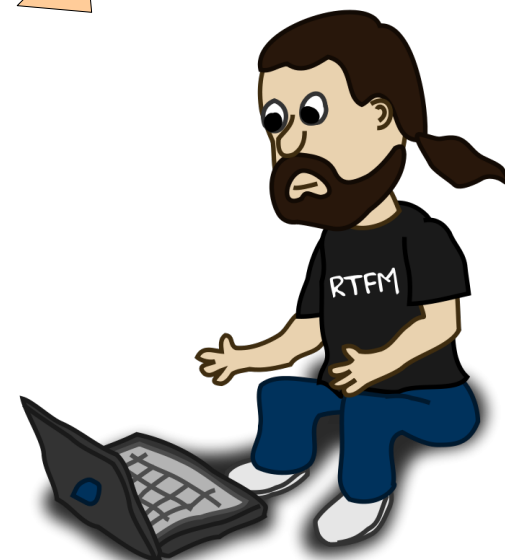
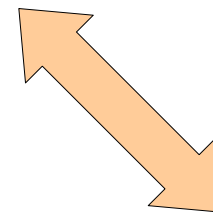
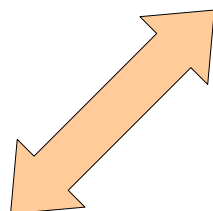
- Tool that does the “copy-your-files” for you
- With the same kind of workflow




Centralized version control

- Tool that does the “copy-your-files” for you
- With the same kind of workflow
- Provides you the goodies for instant action (e.g. diffs, ignored files, blame)
- Usually requires a server (and thus network) for most operations





Centralized version control

- CVS - Concurrent Versions System
- ~~PERFORCE~~
-  SUBVERSION®
- Probably many others





- Provides tools for everything (e.g. find what has changed, when, why and by whom)
- Very quick and easy interface, stays out of your way, encourages granularity and usage
- More efficient storage (both in time and space), encouraging granularity
- Manages merges and conflicts much better than manually



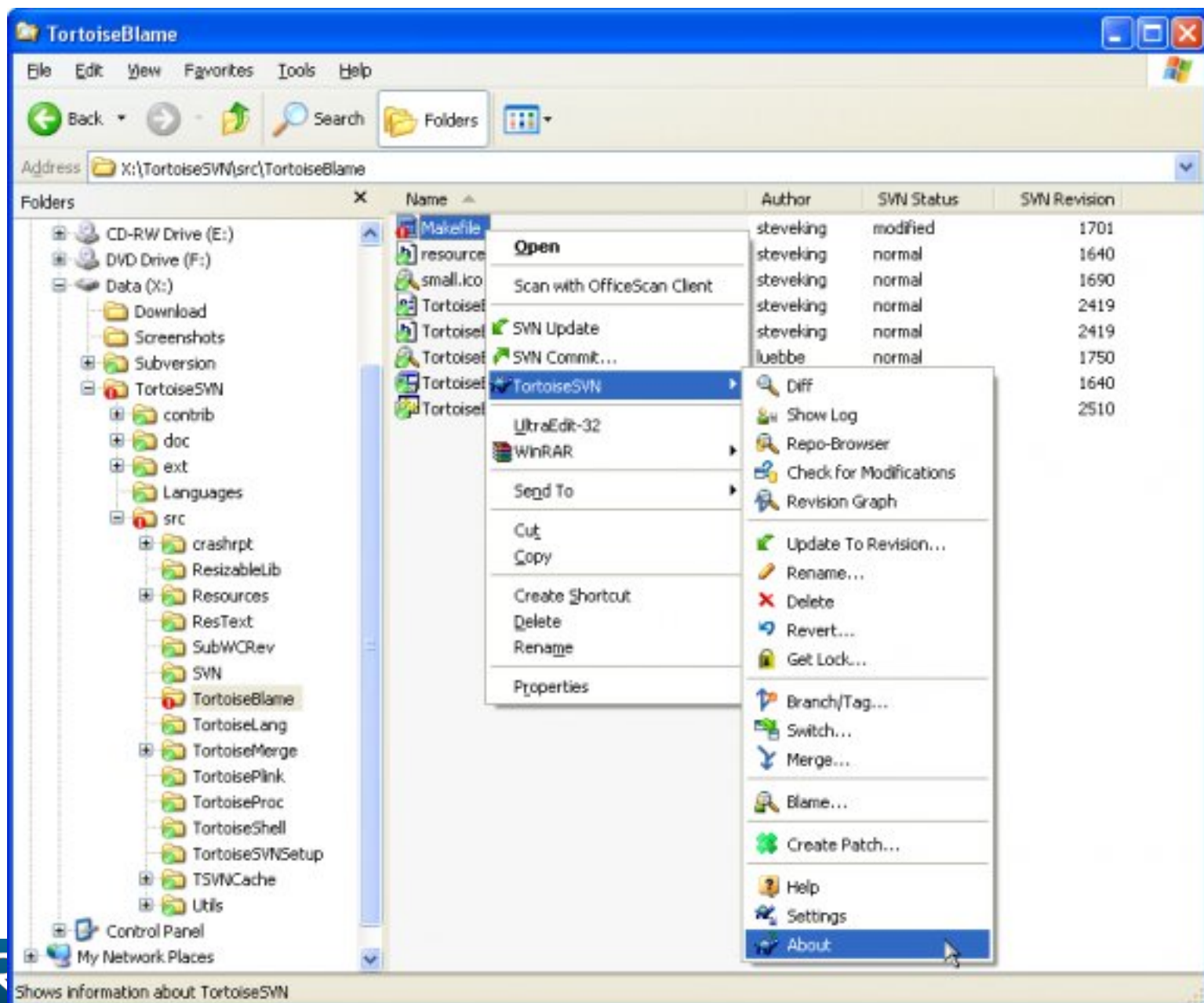
- Command line and GUI similar to shell and File Explorer



```
ddvento@terracotta: ~/subversion.ucar.edu_ddvento/trunk
File Edit View Terminal Help

$ svn status
M      job_memusage.c
$ svn diff
Index: job_memusage.c
=====
--- job_memusage.c      (revision 1353)
+++ job_memusage.c      (working copy)
@@ -17,7 +17,7 @@
  #include <mpi.h>
  #endif

-void printUsage(char **argv) {
+void printUsage(char **argv) { /*new comment*/
  /*      The details options has been removed from the help, because it was
           confusing for the users
           printf("\nUsage:\n%s [--details] <filename-to-run> [<arguments-to-pass>] \n\n", argv[0]);
$ svn mkdir test
A      test
$
```



Typical workflow (centralized)

- 1) Get the latest version
 - `svn update` or `checkout`
- 2) Change your files, check the differences
 - `svn status`
 - `svn diff`
- 3) If not happy, consider reverting changes
 - `svn revert`
- 4) Continue your development in 2) & 3) until happy

Typical workflow (centralized)

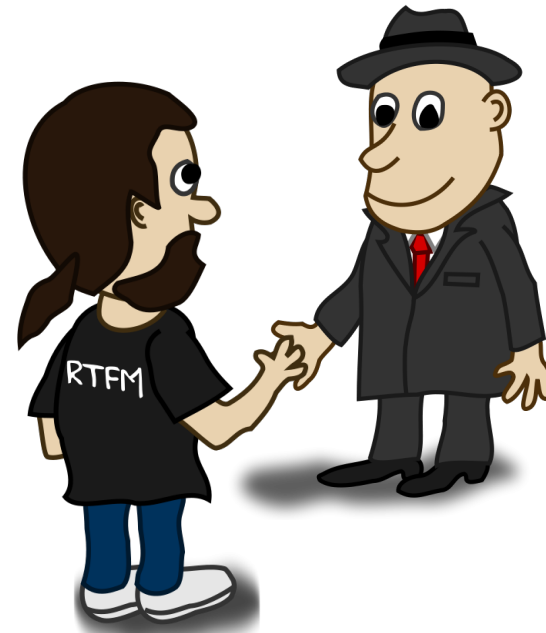
6) Merge with others (if there are others, otherwise skip this)

- `svn update`
- `svn resolve`

7) Test

8) Publish

- `svn commit`



The four W's

- What has been changed in a file? When?
By Whom? `svn blame` / `praise`

```
ddvento@terracotta: ~/subversion.ucar.edu_ddvento/trunk/c/job_memusage
File Edit View Terminal Help
$ svn blame job_memusage.c | head -45 | tail -18
1      ddvento
1      ddvento int main(int argc, char **argv) {
1      ddvento     struct rusage64 us;      /* resource us struct */
13     ddvento     char *runme, space = ' ';
13     ddvento     time_t start_time, stop_time;
1044   ddvento     int detailed = 0, error, exit_status, signal, rank;
24     ddvento
13     ddvento     if (argc > 1) {
13     ddvento         int current_arg = 1;
13     ddvento         if (strcmp(argv[current_arg], "--details") == 0) {
8      ddvento             detailed = 1;
13     ddvento             current_arg++;
1      ddvento         }
22     ddvento         int curr_arg_ct = current_arg, total_size = 0;
22     ddvento
22     ddvento         for(; curr_arg_ct<argc; curr_arg_ct++)
22     ddvento             total_size += strlen(argv[curr_arg_ct]) + 1;
22     ddvento         /* "+ 1" means ' ' or last '/0'
```

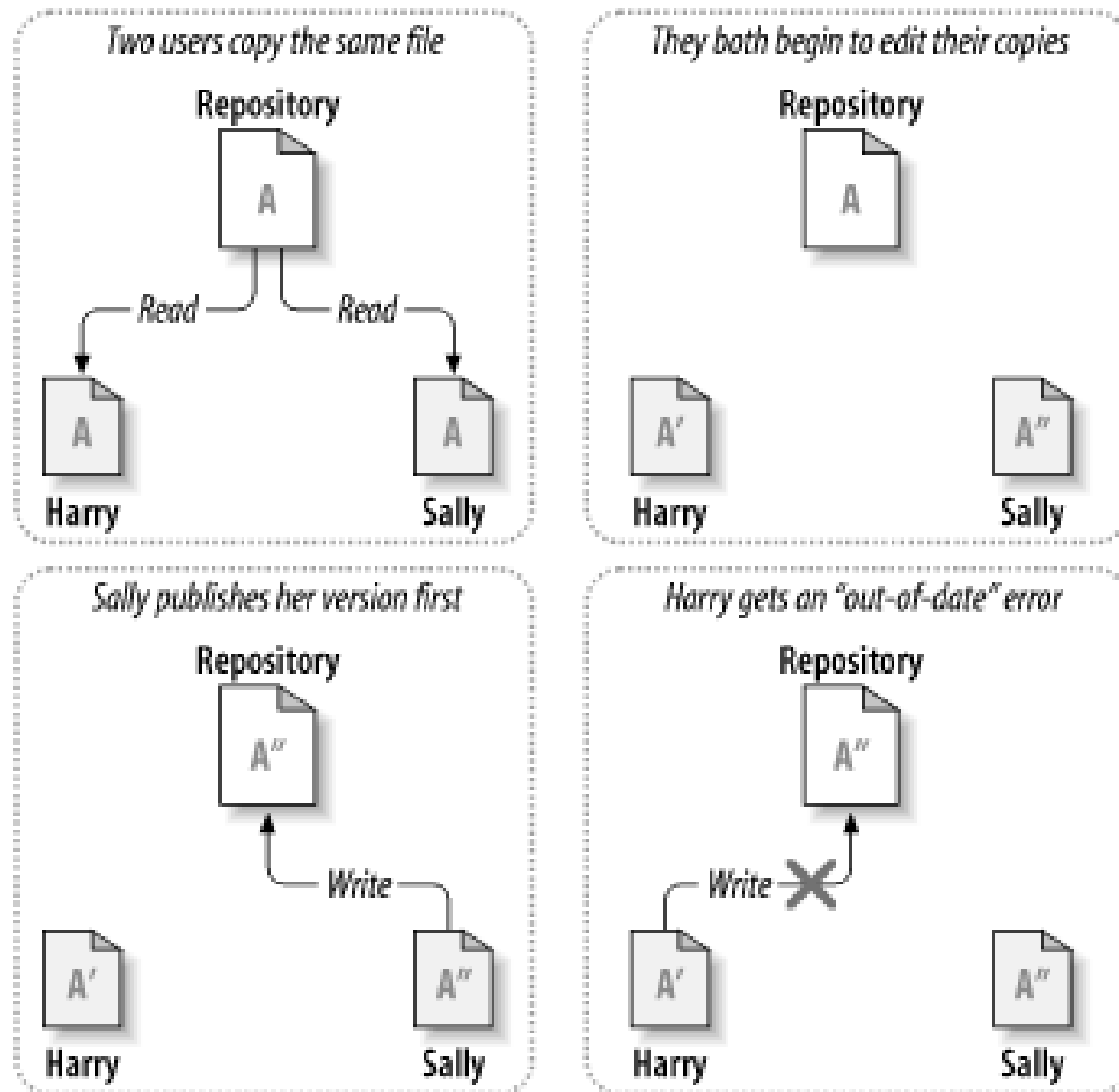
The four W's

- Find Why: `svn log`

```
ddvento@terracotta: ~/subversion.ucar.edu_ddvento/trunk/c/job_memusage
File Edit View Terminal Help
$ svn log job_memusage.c -r 1044
-----
r1044 | ddvento | 2009-12-03 14:45:03 -0700 (Thu, 03 Dec 2009) | 6 lines

The actual program exit status was broken, because it was shifted.
This fixed it, and now it prints also the signal sent to it (if any).
See the observation of this answer for details:
http://stackoverflow.com/questions/774048/774306#774306
-----
$
```

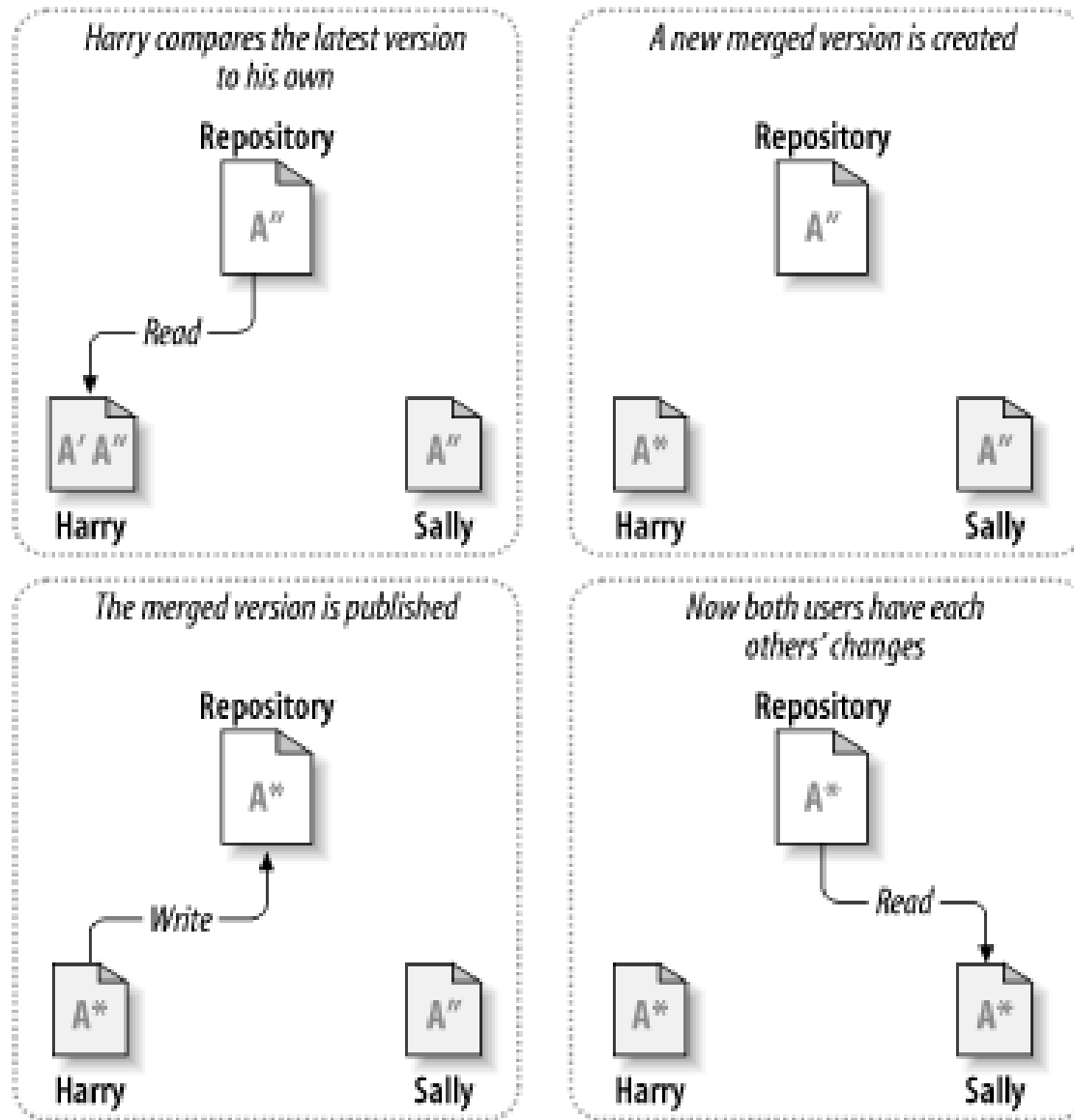
Avoiding Locks (1)



source: subversion “red” book

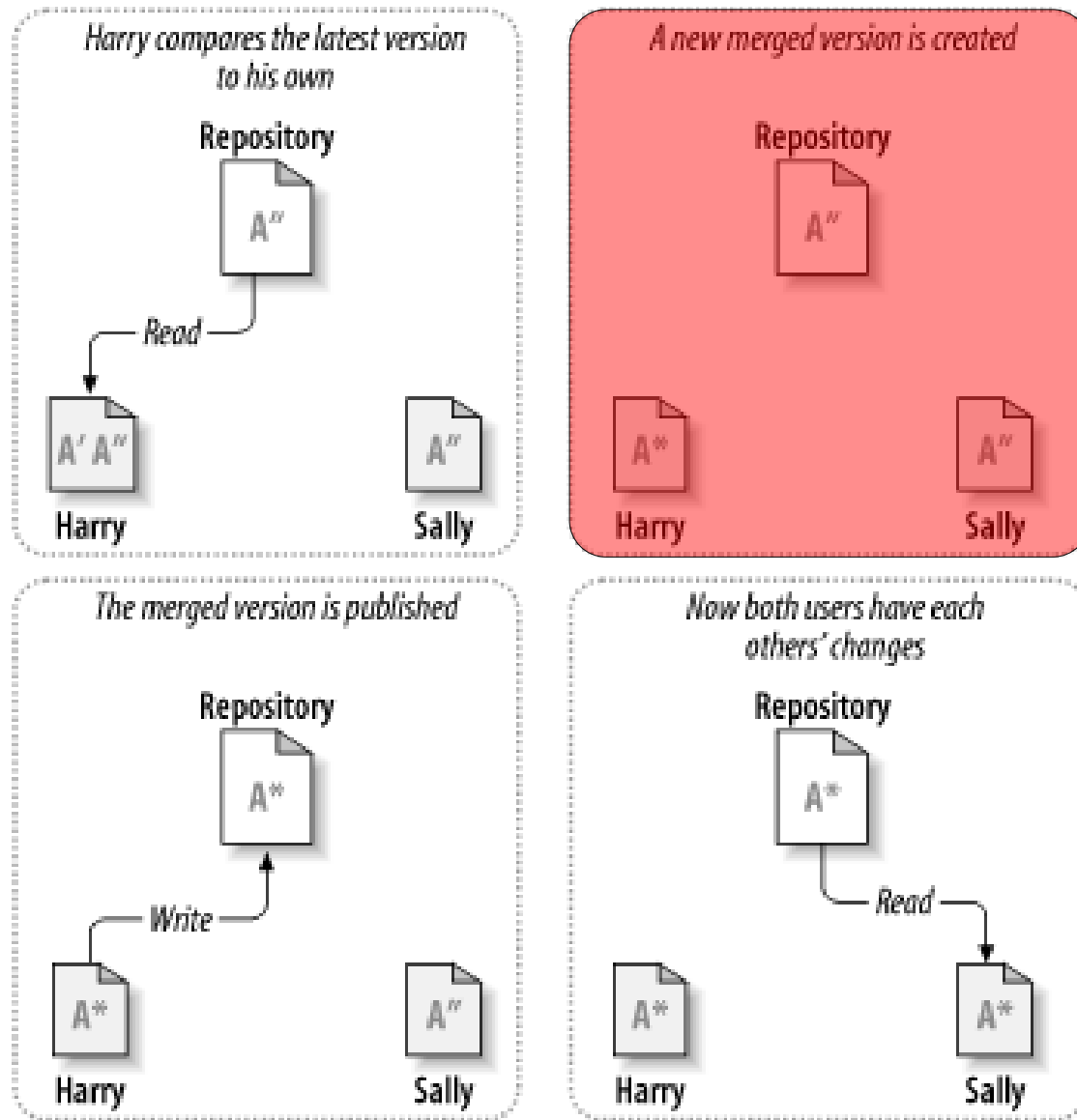


Avoiding Locks (2)



source: subversion “red” book

Avoiding Locks (2)



source: subversion “red” book

Merge

[tmp] mpi5_ugly_arraysum.f90 : [home] mpi5_ugly_arraysum.f90 - Meld

File Edit View Help

Save Undo

[tmp] mpi5_ugly...ly_arraysum.f90

/tmp/tmpe_ddLE-meld/mpi5_ugly_arraysum.f90 Browse...

```
1 program example
2 use mpi ! to be commented out with gfortrant
3 implicit none ! to be commented out with gfortrant
4 integer :: i, first, last, len
5 integer :: split, pieces, rank, ierr, my_first, my_last
6 parameter (len=90)
7 double precision :: a(len), b(len), c(len)
8
9 CALL MPI_Init(ierr)
10 CALL MPI_Comm_size(MPI_COMM_WORLD, pieces, ierr)
11 CALL MPI_Comm_rank(MPI_COMM_WORLD, rank, ierr)
12
13 split = len / pieces
14 my_first = rank * split + 1
15 my_last = (rank + 1) * split
16
17 ! read_data
18 do i=1,len
19   a(i) = 100. * i
20   b(i) = i
21 enddo
22
23 ! do your math in parallel
24 do i=my_first,my_last
25   c(i) = a(i) + b(i)
26   ! this is easy, try matrix multiplication...
27 enddo
28
29 do i=0,len/10-1
30   first = i*10+1
31   last = i*10+10
32   print *, "-"
33   print '("a(",i2,";",i2,")",10(1x,f6.1))',first,last, a(first:last)
34   print '("b(",i2,";",i2,")",10(1x,f6.1))',first,last, b(first:last)
35   print '("c(",i2,";",i2,")",10(1x,f6.1))',first,last, c(first:last)
36 enddo
37 CALL MPI_Finalize(ierr)
38
39 end program example
40
41
42
```

/home/ddvento/subversion.ucar.edu_ddvento/trunk/fortran/parallel_class/MPI Browse...

```
1 program example
2 use mpi ! to be commented out with gfortrant
3 implicit none ! to be commented out with gfortrant
4 integer :: i, first, last, len
5 integer :: split, pieces, rank, ierr, my_first, my_last
6 parameter (len=90)
7 double precision :: a(len), b(len), c(len)
8
9 CALL MPI_Init(ierr)
10 CALL MPI_Comm_size(MPI_COMM_WORLD, pieces, ierr)
11 CALL MPI_Comm_rank(MPI_COMM_WORLD, rank, ierr)
12
13 split = len / pieces
14 my_first = rank * split + 1
15 my_last = (rank + 1) * split
16
17 ! read_data
18 do i=1,len
19   a(i) = 100. * i
20   b(i) = i
21 enddo
22
23 ! do your math in parallel
24 do i=my_first,my_last
25   c(i) = a(i) - b(i) ! subtracting instead of summing
26   ! this is easy, try matrix multiplication...
27 enddo
28
29 do i=0,len/10-1
30   first = i*10+1
31   last = i*10+10
32   print *, "-"
33   print '("a(",i2,";",i2,")",10(1x,f6.1))',first,last, a(first:last)
34   print '("b(",i2,";",i2,")",10(1x,f6.1))',first,last, b(first:last)
35   print '("c(",i2,";",i2,")",10(1x,f6.1))',first,last, c(first:last)
36 enddo
37 CALL MPI_Finalize(ierr)
38
39 end program example
40
41
42
```

INS : Ln 33, Col 1

Practical advice

- 1) Commit often!
- 2) Write short, but useful, specific commit logs
- 3) Try to avoid clutter (e.g. object files, editor temp files, large binaries like MS word doc)
- 4) Do commit everything you do (to a level)
- 5) Do not version external dependencies (e.g. libraries you use), use the `svn:external` property

Other goodies

- Properties can be “attached” to file(s)
svn proplist / propget / propset / propedit
svn:keywords, svn:ignore, svn:externals, svn:needs-lock
- SVN can automagically “create” in your files things like *“This document has been updated on xxx”*
svn:keywords (some are: URL, Author, LastChangedBy, Date, LastChangedDate, Rev, LastChangedRevision, Id)
- Bash completion
- How to make a “release” of your project:
 - create a tag, then
 - svn export

How much to commit?

- As much as you can
- But...
 - 1) If you commit code that doesn't work (or even compile), you may stop others to work
 - 2) Somebody feels ashamed to commit code that is not “clean enough”

Branching and Merging in SVN

- Usually the top level directories in any SVN repository are `tags/`, `branches/` and `trunk/`
- `trunk/` is where development occurs
- `tags/` is where releases are placed
- `branches/` is where “experimental” development occurs

Merge problems in SVN (1)

- Two devs are working in two branches, to avoid stepping on each other's foot
- Eventually they want to merge, and possibly even at intermediate steps
- **Problem #1:** subversion does not track branches properly, i.e. you have to manually remember
 - 1) when (rev id) your branch started and/or
 - 2) when (rev id) you intermediately merged

Merge problems in SVN (2)

- **Problem #2:** `svn merge` is just a 2-way diff between two versions, i.e.
 - 1) the information in the common ancestor is not stored (nor used)
 - 2) lot of “irrelevant” conflicts show up, especially (but not only) if you didn't correctly tracked rev-id (as said in previous slide)
- Bottom line: most people try branching/merging on SVN and then say “This is broken, never again!”

subversion pros/cons

- ✓ Very easy to use
- ✓ Natural workflow
- ✓ Get rid of the lock-modify-unlock model
- ✓ Powerful commands for lot of things
- ✓ Big improvements compared to CVS, which SVN tried to displace

subversion pros/cons

- ✓ Very easy to use
- ✗ Doesn't fully work offline, or when the server is down
- ✓ Natural workflow
- ✗ trunk/ is at high risk of breaking
- ✗ To avoid breaking trunk/ you may use branch/ (painful to merge back)
- ✗ or avoid committing altogether (losing version control functionality when you need it most)
- ✓ Big improvements compared to CVS, which it tried to displace

Distributed version control

- Split the “commit” in two:
 - 1) Remember this point in time (for myself, for now only within this working copy)

Distributed version control

- Split the “commit” in two:
 - 1) Remember this point in time (or myself, for now only within this working copy)
 - 2) Make it visible to others (or myself, but outside this working copy)



Distributed version control

- Benefits:
 - 1) works locally, no network or repository needed
 - 2) further encourage granularity
 - 3) more information is kept, making merging better!

Distributed version control

- Three main contenders:

- GIT



- Mercurial



- Bazaar



- And many minor ones, e.g. DARCS



Choices, choices, choices

- You have (at least) 3 choices for the distributed version control tool to use
- Each one of them support many (if not several) workflows

Choices, choices, choices

- You have (at least) 3 choices for the distributed version control tool to use
- Each one of (several) workflows



Choices, choices, choices

- You have (at least) 3 choices for the distributed version control tool to use
- Each one of them support many (if not several) workflows
- Pick one, and use it, don't be let you down by the “too many choices” problem

A possible distributed workflow

- 1) Get the latest version
- 2) Change your files
- 3) Commit often (locally!!)
- 4) If not happy, consider reverting changes
- 5) If not happy, goto 2)
- 6) Merge
- 7) Test
- 8) Publish

Other workflows

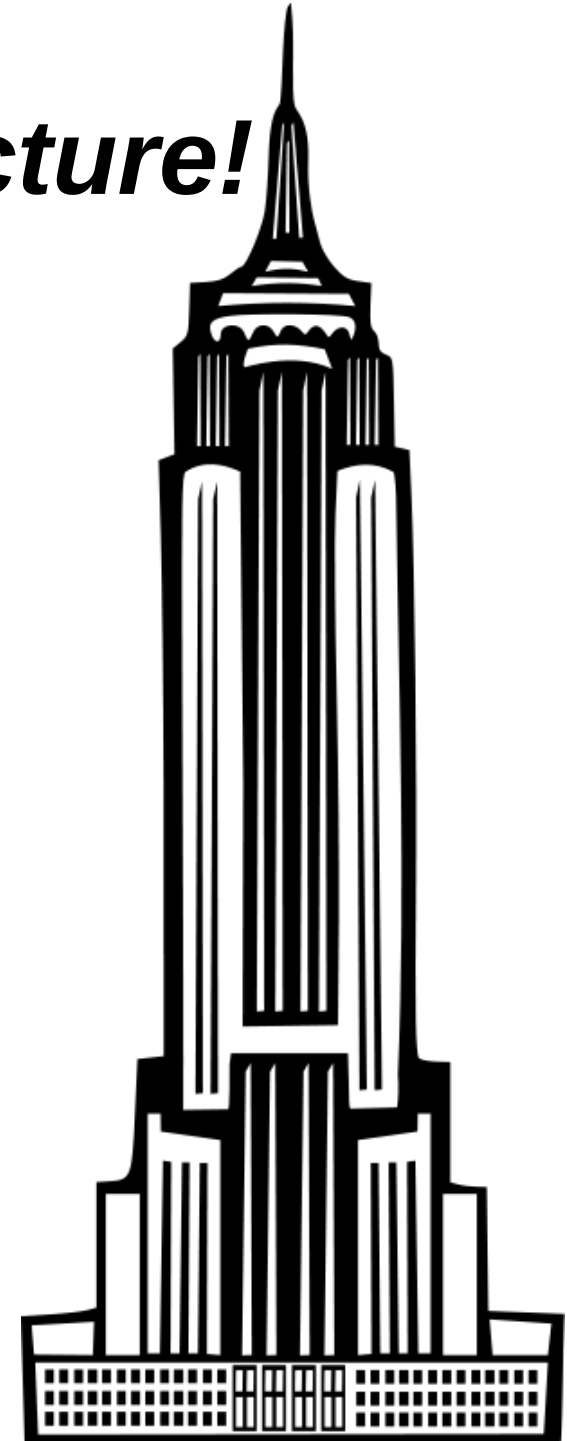
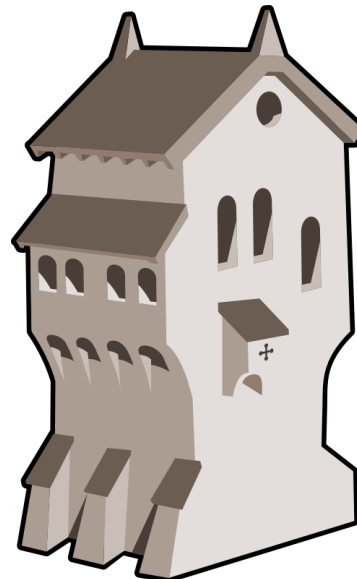
- <http://doc.bazaar.canonical.com/bzr.dev/en/user-guide/index.html>
- <http://wiki.bazaar.canonical.com/Workflows>
- <http://osteele.com/archives/2008/05/my-git-workflow>
- <http://tomayko.com/writings/the-thing-about-git>
- <http://mercurial.selenic.com/wiki/Workflows>

Merging done right

Merging done right

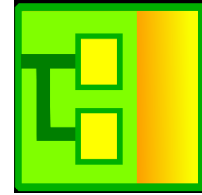
- All the distributed VCS are able to properly manage a situation like this:
 - I change a function a little bit
 - Another developer move it somewhere else and does some other changes
- However they have different definitions of “right”
- SVN doesn’t track properly these steps, so when it merges, it thinks a new function just showed up, and does the “wrong” thing

Main difference: architecture!



Main difference: architecture!

- GIT is a user-side filesystem



- HG is like Japanese streets



- BZR is more like the old style (looking right)



Side-by-Side Comparison

- In the next few slides I'll show some tables
- Comparing features and meta-features
- Why care about the meta-features?
- Because they are a measure of popularity, and popularity matters

Architecture

VCS	type	history model	Preferred branching model	merge	bug/feature finding
svn	revision	snapshot (*)	Completely broken	2-way	Fully manual
bzr	files	snapshot	flexible	3-way, LCA, octopus	bzr bisect (plugin)
hg	files	changeset	Clone + pull (or push) + merge	3-way	hg bisect (assisted)
git	content tracking (**)	snapshot	explicitly named branches	3-way, recursive, octopus, custom	git bisect

(*) internally it stores changesets, but it exposes only snapshots

(**) userland filesystem (inodes, etc)

Architecture and tracking issues

VC S	renaming tracking	directory tracking	partial tree checkout	partial tree commit
svn	yes (copy+delete, changes do not follow copies)	yes	yes	yes
bzr	yes	yes (directories are 1 st class objects empty directories possible, renames are fully tracked)	no (under developmt - plugin)	yes (and shelve)
hg	yes (copy+delete, changes do follow copies)	no (no empty directories allowed)	no (under developmt)	no
git	not explicitly	partial (no empty directories allowed)	sparse checkout (***)	staging area

(***) not really – you still need to clone the whole repository

GUI and Eclipse

VCS	GUI frontends	Eclipse plugin
svn	many, good	excellent
bzr	a few, good	poor
hg	one, good	good
git	many, fair	poor

VCS	hosting	open source	closed source projects allowed	bug tracker, wiki, forum, stats	code review
svn	Google Project Hosting	no	no	yes	yes
	sourceforge	no	no	yes	no
	savannah	yes	no	some	no
bzr	launchpad	yes	yes, paid	yes	yes
	savannah	yes	no	some	no
hg	bitbucket	no	yes, free for small projects	yes	no
	Google Project Hosting	no	no	yes	yes
	sourceforge	no	no	yes	no
	savannah	yes	no	some	no
git	github	no	yes, paid	yes	yes
	gitorious	yes	yes, paid	yes	yes
	sourceforge	no	no	yes	no
	savannah	yes	no	some	no

Bazaar pros/cons

- ✓ Easy to use
- ✓ Slightly more flexible than the others
- ✓ Slightly faster than Mercurial
- ✓ Partial commits (subset of files or directories, like SVN) – shelve/unshelve another option
- ✓ Octopus merging (like GIT, unlike HG)
- ✓ Only DVCS to have true renaming tracking (however other models works too)
- ✓ Great default cross-platform GUI
- ✗ Slower than git (but how large is your project?)
- ✗ Canonical stigma (and large role in it)
- ✗ Lower number of high-level projects using it

Mercurial pros/cons

- ✓ Very easy to use
 - ✓ Less flexible than others (can be a con)
 - ✓ Only one to have free (== unpaid) hosting for closed source projects
 - ✓ Good Eclipse plugin
-
- ✗ Slowest (but how large is your project?)
 - ✗ No partial commits (annoyance more than a con)
 - ✗ Less flexible than others (can be a pro)

GIT pros/cons

- ✓ Complex and flexible (can be a con)
- ✓ Extremely fast, especially for large projects (the core is written in C)
- ✓ Nice branching model (including, but not limited to the staging area)
- ✓ GitHub
- ✗ Many GIT commands are shell or perl scripts, sometimes low quality (changing since git 1.6)
- ✗ Complex and flexible (can be a pro) – GIT has the largest command set of the three
- ✗ When in doubt, do exactly the opposite of CVS (it was supposed to be a pro!)
- ✗ Repository requires manual “repacks” of metadata

Last word on comparison

- Via fast-export and fast-import everything can be converted to everything else, without losing history
- They are catching each others
- Like driving a car, different brands put switches in different locations, but you should be able to drive any without too much efforts

Choices

- So pick one, and use it, don't be let you down by the “too many choices” problem





References (SVN)

- Always handy:
svn help / svn help xxx

- Subversion Home
<http://subversion.apache.org/>



- Subversion “red” book
<http://svnbook.red-bean.com/>

- Subversion GUI frontends
<http://www.rapidsvn.org/> (cross-platform)



<http://scplugin.tigris.org/> (MacOS)



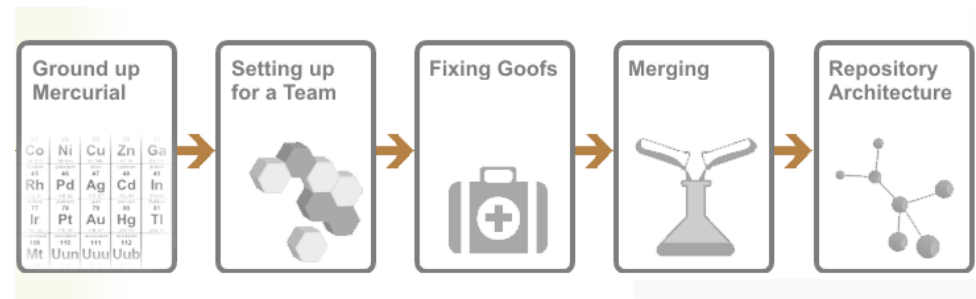
<http://tortoisesvn.tigris.org/> (Windows)

References (HG)

- Always handy:
`hg help / hg help xxx`
- Mercurial home
<http://mercurial.selenic.com/>



- Good place to start
<http://www.hginit.com>



- The Definitive Guide
<http://hgbook.red-bean.com/>



- Mercurial GUI frontends
<http://tortoisehg.bitbucket.org/> (cross-platform)
<https://bitbucket.org/snej/murky/> (MacOS only)

References (BZR)

- Always handy:
`bzr help / bzr help xxx`
- Bazaar home
<http://bazaar.canonical.com>
- Good place to started
<http://doc.bazaar.canonical.com/latest/en/mini-tutorial/>
- Bazaar GUI frontends (cross-platform)
<https://launchpad.net/bzr-explorer> (official)
<https://launchpad.net/qbzr/> (Qt)
<https://launchpad.net/tortoisebzr/> (windows only)



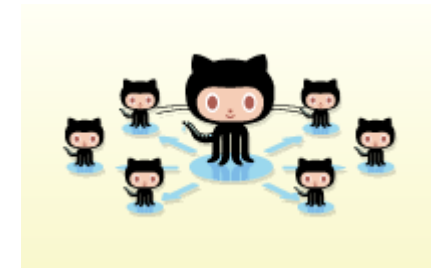
References (GIT)

- Git home
<http://git-scm.com/>



- Good place to understand under the hood
<http://tom.preston-werner.com/2009/05/19/the-git-parable.html>

- Great, simple, getting started
<http://gitref.org/>



- Other stuff worth knowing
<https://git.wiki.kernel.org/index.php/GitSvnCrashCourse>
<http://www-cs-students.stanford.edu/~blynn/gitmagic>
<https://sea.ucar.edu/event/unlocking-secrets-git>

References (comparison)

- Biased comparison of HG, BZR and GIT (good to read what advocates of a tool say)

<http://hgbook.red-bean.com/read/how-did-we-get-here.html> (HG-GIT)

<http://whygitisbetterthanx.com/>

<http://doc.bazaar.canonical.com/migration/en/why-switch-to-bazaar.html>

- Unbiased comparison of HG, BZR and GIT

http://en.wikipedia.org/wiki/Comparison_of_revision_control_software

<http://rg03.wordpress.com/2009/04/07/mercurial-vs-git/>

<http://automatthias.wordpress.com/2007/06/07/directory-renaming-in-scm/>





Attribution-Noncommercial-Share Alike 3.0 United States

You are free:



to Share — to copy, distribute, display, and perform the work



to Remix — to make derivative works

Under the following conditions:



Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Noncommercial. You may not use this work for commercial purposes.



Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.
- Any of the above conditions can be waived if you get permission from the copyright holder.
- Apart from the remix rights granted under this license, nothing in this license impairs or restricts the author's moral rights.

Disclaimer

Your fair use and other rights are in no way affected by the above.
This is a human-readable summary of the [Legal Code](http://creativecommons.org/licenses/by-nc-sa/3.0/us/) (the full license).

<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>

Davide Del Vento