Introduction to version control

David Rey – DREAM

Overview

- Collaborative work and version control
- CVS vs. SVN
- Main CVS/SVN user commands
- Advanced use of CVS/SVN

Overview

- Collaborative work and version control
- CVS vs. SVN
- Main CVS user commands
- Advanced use of CVS

Collaborative work and version control: examples

- Development
  - Source files: C, C++, java, Fortran, Tcl, Python, shell scripts, …
  - Build/config files: Makefile, ant, …
- Text documents/articles/bibliographies
  - Plain text
  - Latex/bibtex
- Web pages
  - Html
  - Php, javascripts, …
- XML documents
- ...

A software development process at INRIA

- INRIA recommandations about software development:
  - http://www-sop.inria.fr/dream/rapports/devprocess/index.html
  - http://www-sop.inria.fr/dream/rapports/devprocess/main005.html#toc8

- « Best practices »:
  - CVS: http://www.tldp.org/REF/CVS-BestPractices/html/index.html
  - SVN: http://svn.collab.net/repos/svn/trunk/doc/user/svn-best-practices.html

Version control: main ideas

- Distributed documents/collaborative work
  - Automatic merging
  - Alarms on conflicts
  - Easy communication between users (log messages, emails, …)
- Version control: incremental versions
  - All previous versions available
  - Minimal necessary disk space (incremental)
  - History of changes/logs

Version control software

- CVS: http://ximbiot.com/cvs/
  - TortoiseCVS (http://www.tortoisecvs.org/)
  - WinCVS (http://www.wincvs.org/)
  - ...
- Subversion (SVN): http://subversion.tigris.org/
  - TortoiseSVN (http://tortoisesvn.tigris.org/)
  - ...
- Forges that use version control software:
  - GForge: www.gforge.org, typically the Inria gforge: gforge.inria.fr
  - Savannah: savannah.gnu.org
  - LibreSource: libresource.inria.fr
  - Visual Source Safe (pour Windows): msdn.microsoft.com/vstudio/previous/ssafe

CVS: Concurrent Versions System
SVN: Subversion

- Widely used on a lot of different platforms (Linux, Windows, …)
- Open source license
- For local and/or distant usage
- Recommended for INRIA software developments
- Several possible clients
  - Command line
  - GUI: tkCVS, jCVS, WebCvs, Jalindi igloo, WinCVS, TortoiseCVS, TortoiseSVN, …

What CVS/SVN is for ?

- Several users work together at the same time on the same files (concurrency)

- Version control
  - Tags
  - Version comparisons
  - Multiple lines of development in the same code base
  - Branching
  - Tolerates binary files
  - Event control (e.g., notification)

What CVS/SVN is not for ?

- Backup
- Bug tracking
- Source documentation
- Dependencies
- …

Overview

- Collaborative work and version control
- CVS vs. SVN
- Main CVS/SVN user commands
- Advanced use of CVS/SVN

SVN = CVS (++)

- SVN is for the same usage as CVS
- 99% of SVN commands are the same as CVS (on the client side)
- SVN seems to be very similar to CVS with more intuitive behavior:
  - Add/delete/move for files and folders are automatically taken into account
  - Recursive add into folders is automatically done (simpler than import command under CVS)
  - When a commit is done on a subset of the files, the whole project version number is incremented
  - Binary files are tolerated by default
  - *svn update* automatically downloads new folders and files

Want to change from CVS to SVN

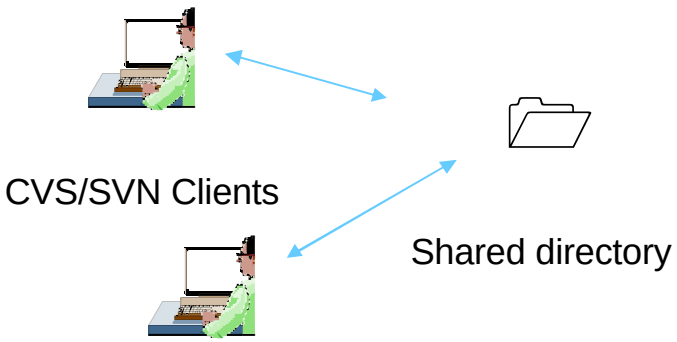
- I would like to use SVN, but my project is already under CVS…
- On the other hand, SVN seems easier to use…
- There is a tool to convert a whole CVS repository easily (inserting the history of files) into a SVN base:
  - cvs2svn: http://cvs2svn.tigris.org/
  - For more details: http://siteadmin.gforge.inria.fr/FAQ.html#Q3pre

Overview

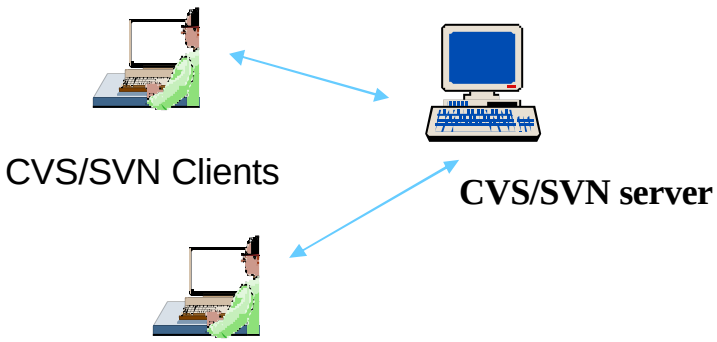
- Collaborative work and version control
- CVS vs. SVN
- Main CVS/SVN user commands
- Advanced use of CVS/SVN

CVS/SVN: client/server architecture

- 2 possibilities:



Local

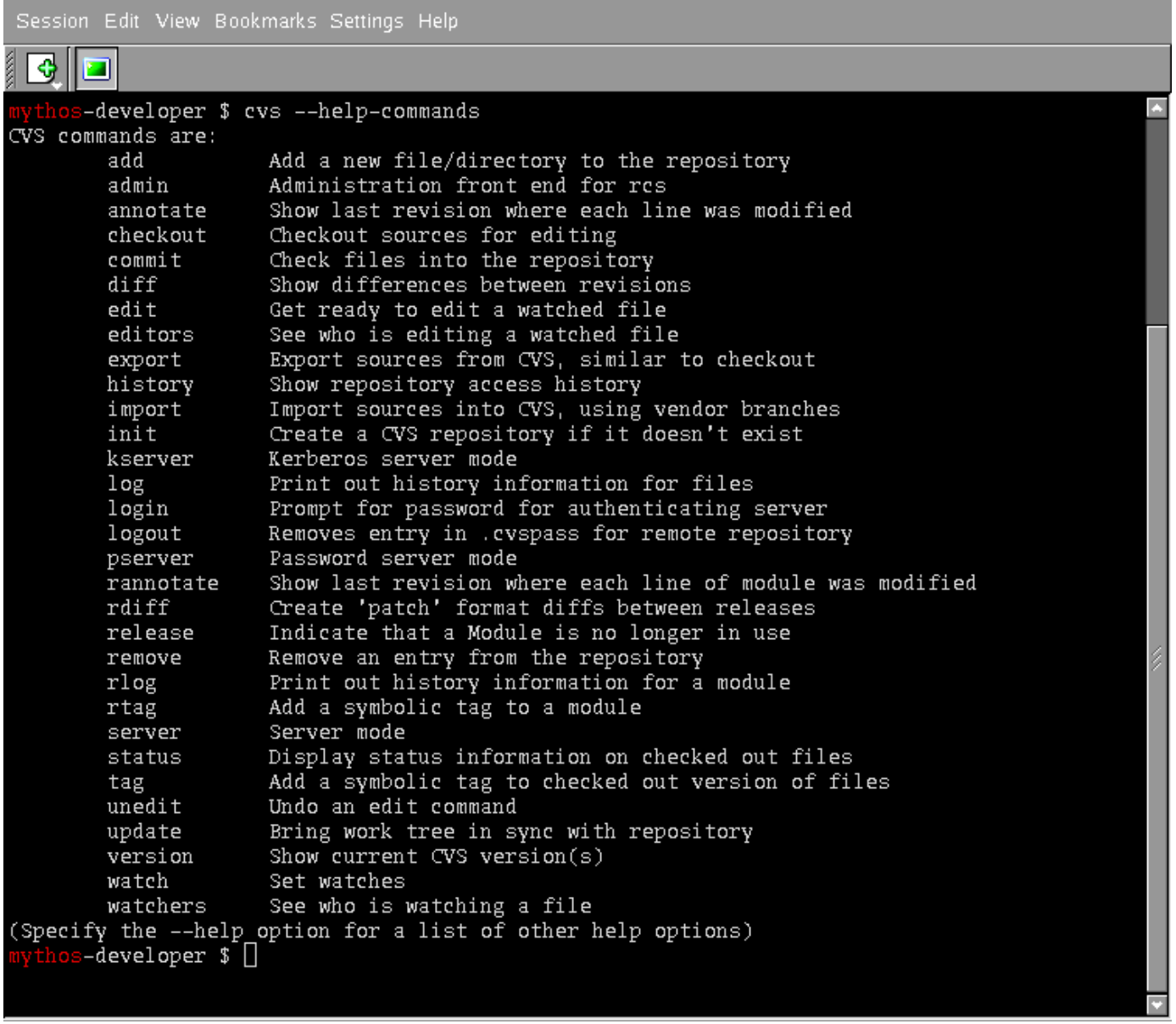


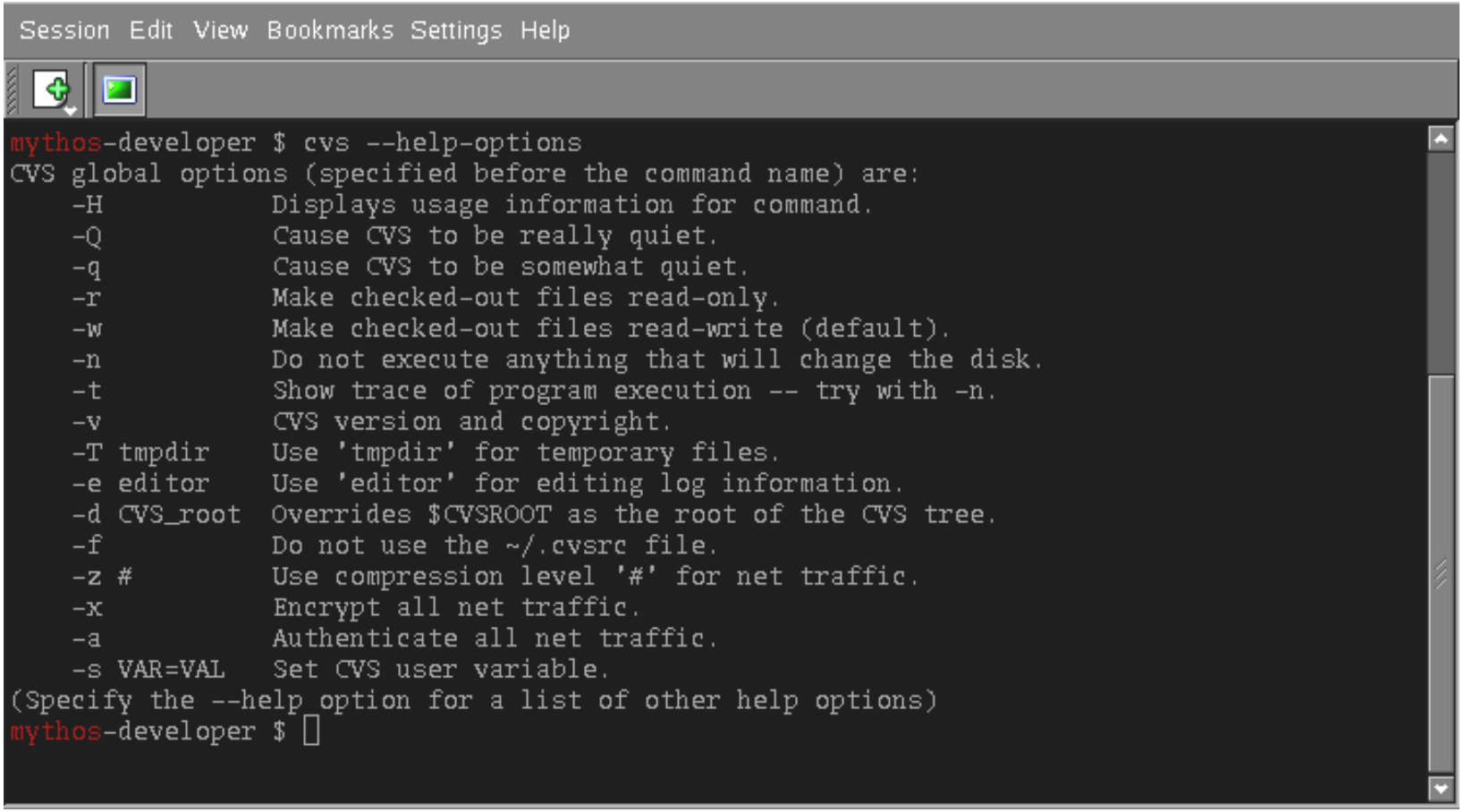
Client/Server

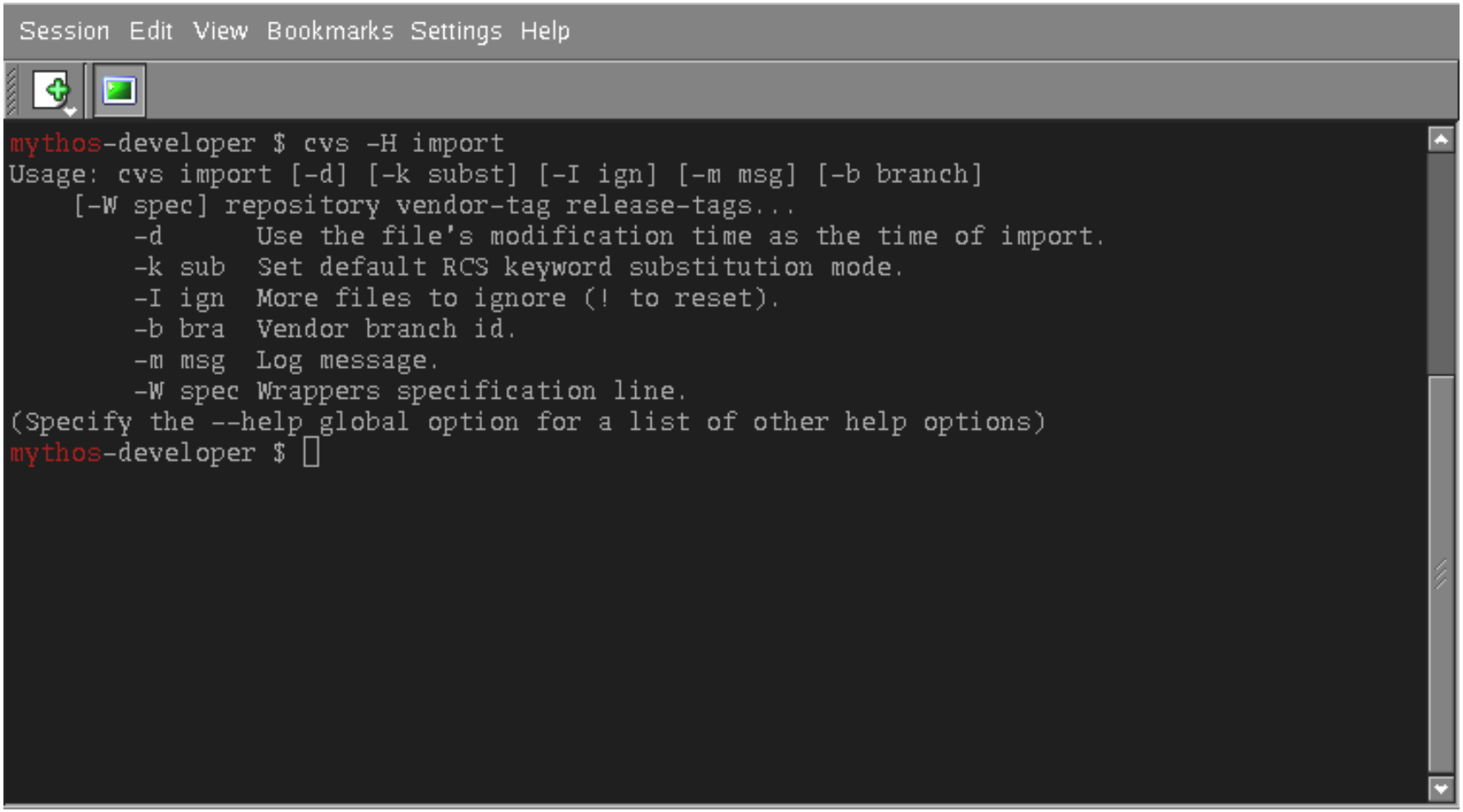
CVS help

- `cvs --help-commands` (lists available commands)
- `cvs --help-options` (lists general options)
- `cvs -H import` (specific help for a command and its specific options, e.g. “import”)

- → CVS general options + specific command options are different :
  - `cvs [general options] <command> [specific options] [args]`

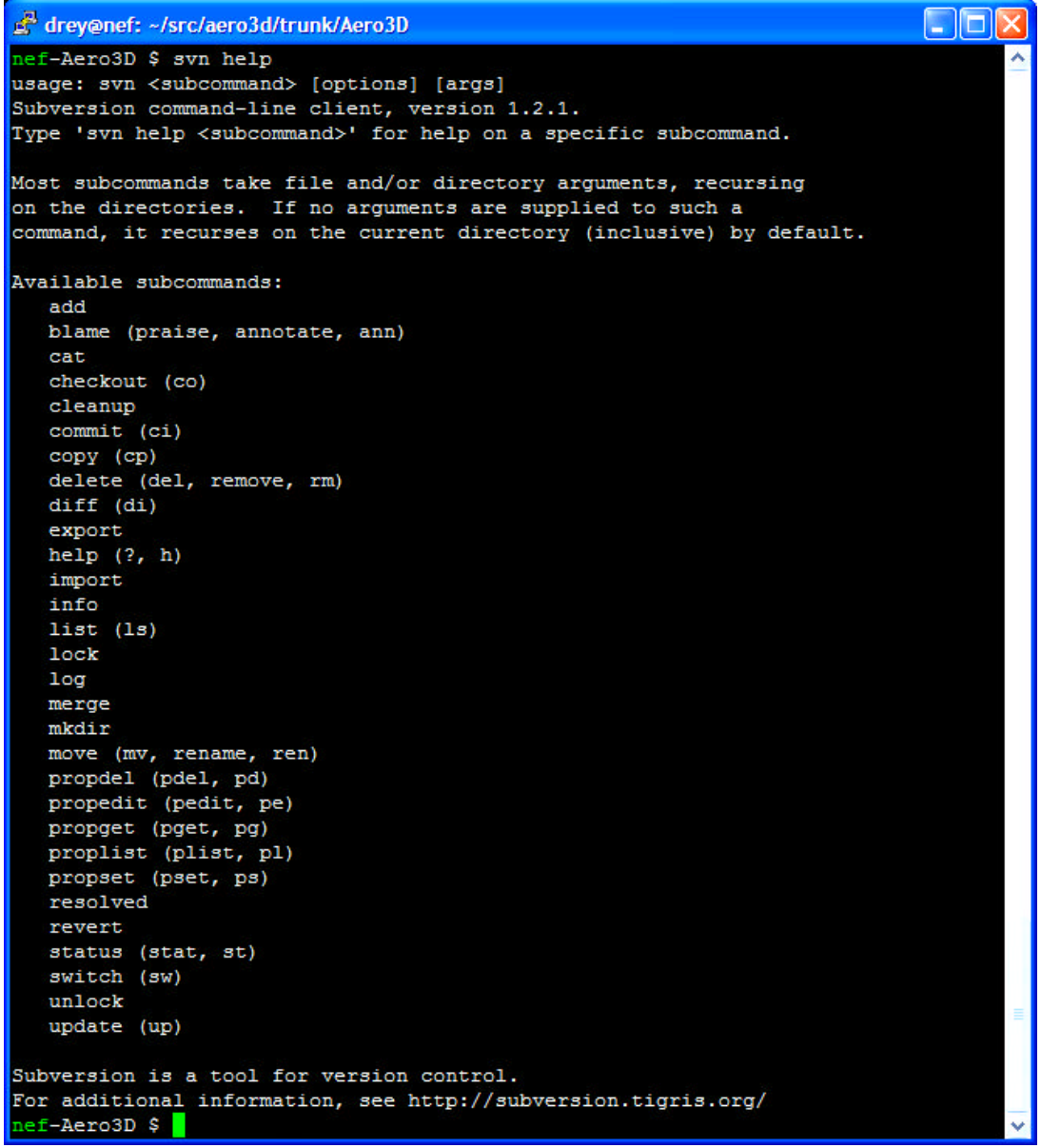

Session Edit View Bookmarks Settings Help
mythos-developer $ cvs --help-commands
CVS commands are:
add Add a new file/directory to the repository
admin Administration front end for rcs
annotate Show last revision where each line was modified
checkout Checkout sources for editing
commit Check files into the repository
diff Show differences between revisions
edit Get ready to edit a watched file
editors See who is editing a watched file
export Export sources from CVS, similar to checkout
history Show repository access history
import Import sources into CVS, using vendor branches
init Create a CVS repository if it doesn't exist
kserver Kerberos server mode
log Print out history information for files
login Prompt for password for authenticating server
logout Removes entry in .cvspass for remote repository
pserver Password server mode
rannotate Show last revision where each line of module was modified
rdiff Create 'patch' format diffs between releases
release Indicate that a Module is no longer in use
remove Remove an entry from the repository
rlog Print out history information for a module
rtag Add a symbolic tag to a module
server Server mode
status Display status information on checked out files
tag Add a symbolic tag to checked out version of files
unedit Undo an edit command
update Bring work tree in sync with repository
version Show current CVS version(s)
watch Set watches
watchers See who is watching a file
(Specify the --help option for a list of other help options)
mythos-developer $

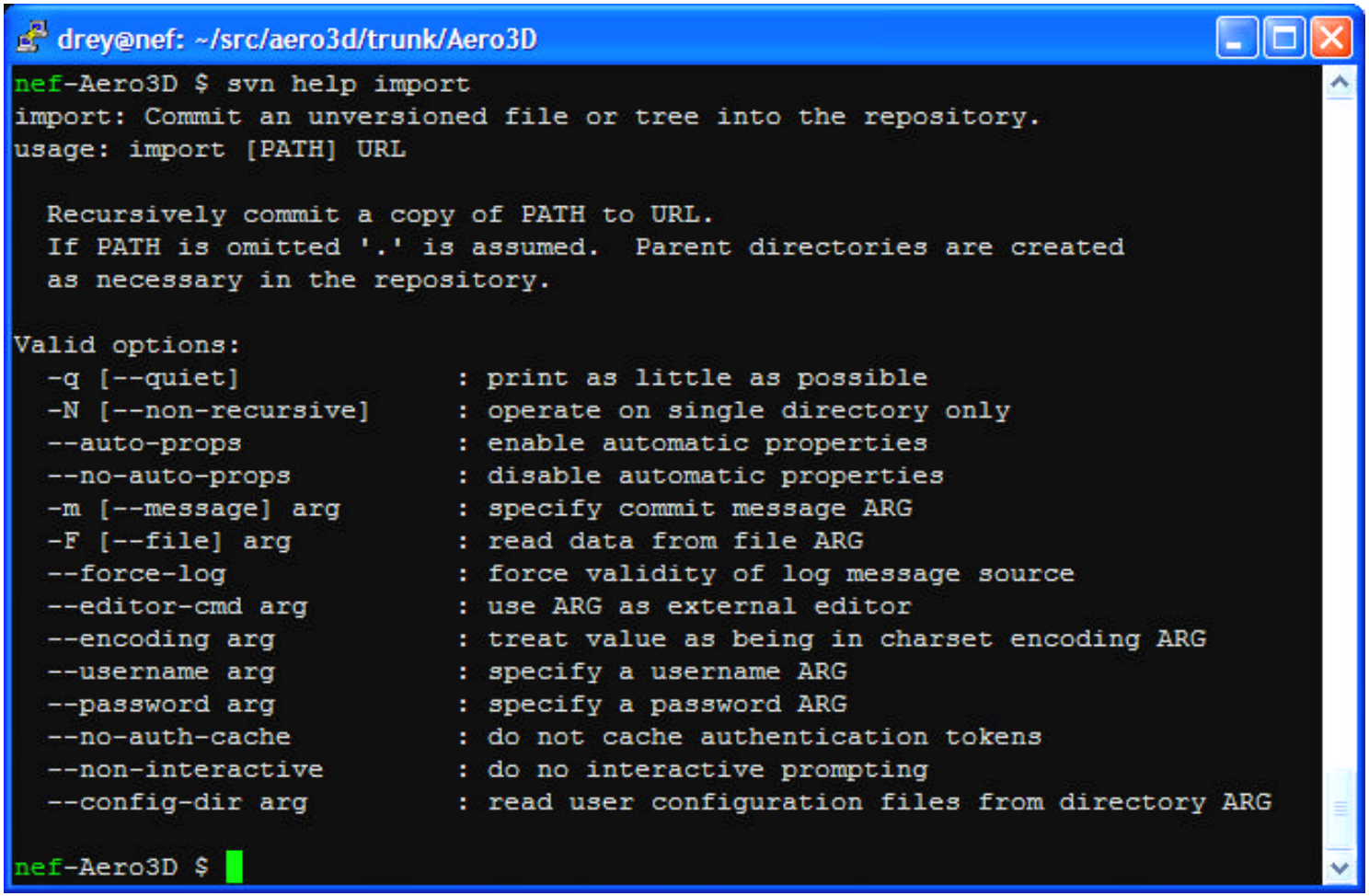

Session Edit View Bookmarks Settings Help
mythos-developer $ cvs --help-options
CVS global options (specified before the command name) are:
-H Displays usage information for command.
-Q Cause CVS to be really quiet.
-q Cause CVS to be somewhat quiet.
-r Make checked-out files read-only.
-w Make checked-out files read-write (default).
-n Do not execute anything that will change the disk.
-t Show trace of program execution -- try with -n.
-v CVS version and copyright.
-T tmpdir Use 'tmpdir' for temporary files.
-e editor Use 'editor' for editing log information.
-d CVS_root Overrides $CVSROOT as the root of the CVS tree.
-f Do not use the ~/.cvsrc file.
-z # Use compression level '#' for net traffic.
-x Encrypt all net traffic.
-a Authenticate all net traffic.
-s VAR=VAL Set CVS user variable.
(Specify the --help option for a list of other help options)
mythos-developer $


Session Edit View Bookmarks Settings Help
mythos-developer $ cvs -H import
Usage: cvs import [-d] [-k subst] [-I ign] [-m msg] [-b branch]
[-W spec] repository vendor-tag release-tags...
-d Use the file's modification time as the time of import.
-k sub Set default RCS keyword substitution mode.
-I ign More files to ignore (! to reset).
-b bra Vendor branch id.
-m msg Log message.
-W spec Wrappers specification line.
(Specify the --help global option for a list of other help options)
mythos-developer $

SVN help

- `svn help` (lists available commands)
- `svn help import` (specific help for a command and its specific options, e.g. “import”)
- → SVN command line:
  - `svn <subcommand> [options] [args]`


drey@nef: ~/src/aero3d/trunk/Aero3D
nef-Aero3D $ svn help
usage: svn <subcommand> [options] [args]
Subversion command-line client, version 1.2.1.
Type 'svn help <subcommand>' for help on a specific subcommand.
Most subcommands take file and/or directory arguments, recursing
on the directories. If no arguments are supplied to such a
command, it recurses on the current directory (inclusive) by default.
Available subcommands:
add
blame (praise, annotate, ann)
cat
checkout (co)
cleanup
commit (ci)
copy (cp)
delete (del, remove, rm)
diff (di)
export
help (?, h)
import
info
list (ls)
lock
log
merge
mkdir
move (mv, rename, ren)
propdel (pdel, pd)
propedit (pedit, pe)
propget (pget, pg)
proplist (plist, pl)
propset (pset, ps)
resolved
revert
status (stat, st)
switch (sw)
unlock
update (up)
Subversion is a tool for version control.
For additional information, see http://subversion.tigris.org/
nef-Aero3D $


drey@nef: ~/src/aero3d/trunk/Aero3D
nef-Aero3D $ svn help import
import: Commit an unversioned file or tree into the repository.
usage: import [PATH] URL
Recursively commit a copy of PATH to URL.
If PATH is omitted '.' is assumed. Parent directories are created
as necessary in the repository.
Valid options:
-q [--quiet] : print as little as possible
-N [--non-recursive] : operate on single directory only
--auto-props : enable automatic properties
--no-auto-props : disable automatic properties
-m [--message] arg : specify commit message ARG
-F [--file] arg : read data from file ARG
--force-log : force validity of log message source
--editor-cmd arg : use ARG as external editor
--encoding arg : treat value as being in charset encoding ARG
--username arg : specify a username ARG
--password arg : specify a password ARG
--no-auth-cache : do not cache authentication tokens
--non-interactive : do no interactive prompting
--config-dir arg : read user configuration files from directory ARG
nef-Aero3D $

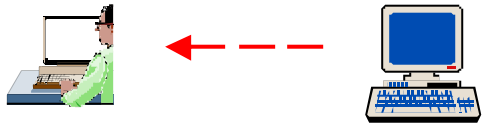
Import

- Used to synchronize a local tree as a whole with the server
- Most often used to initialize the server tree
- Can also be used after this initial step at any time (with CVS, that avoids one-by-one addition in the case of multiple files)

```
cd local_directory
cvs import –m “message” directory_on_base branch_name version_name
```

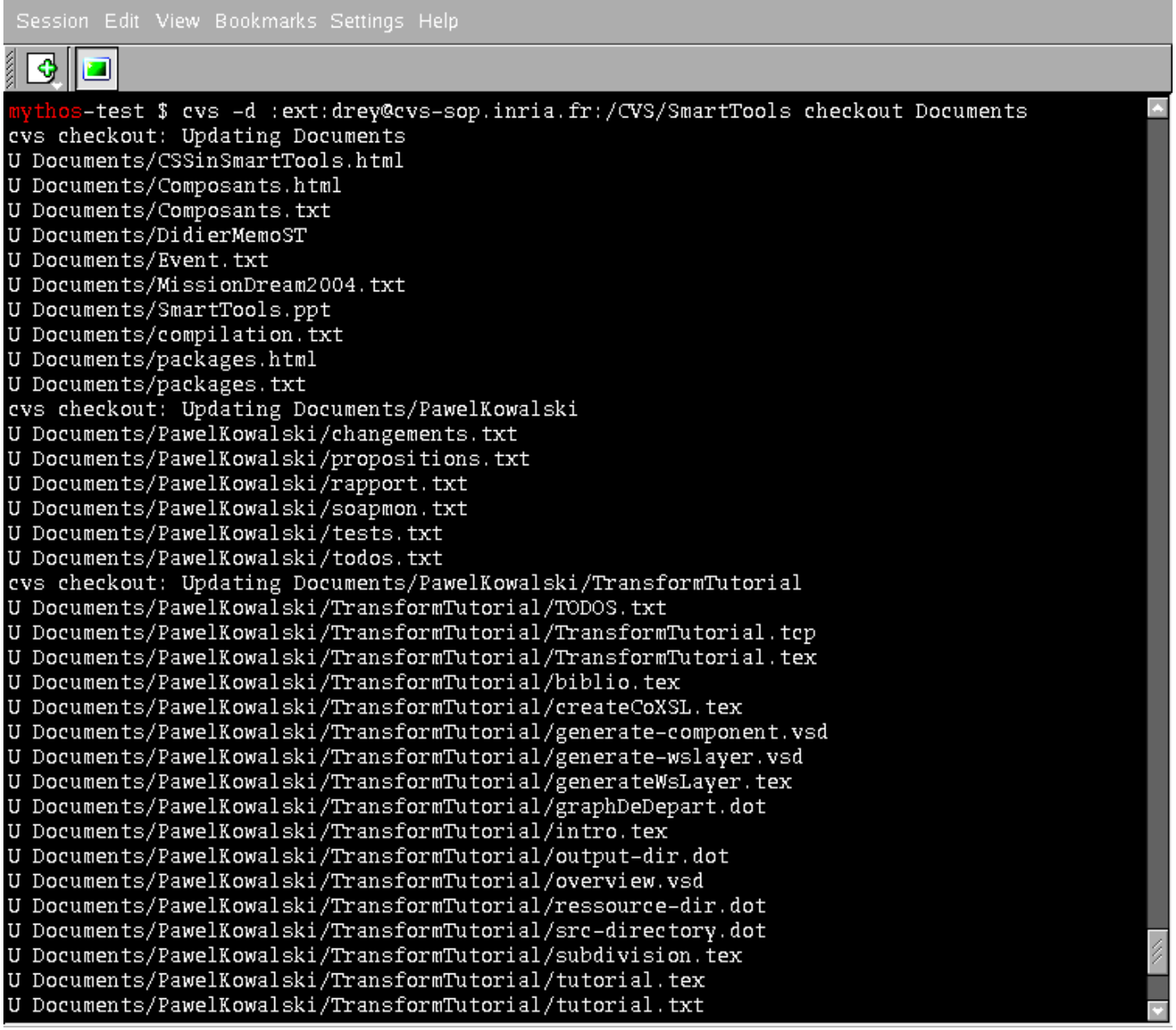
```
svn import [PATH] URL
```

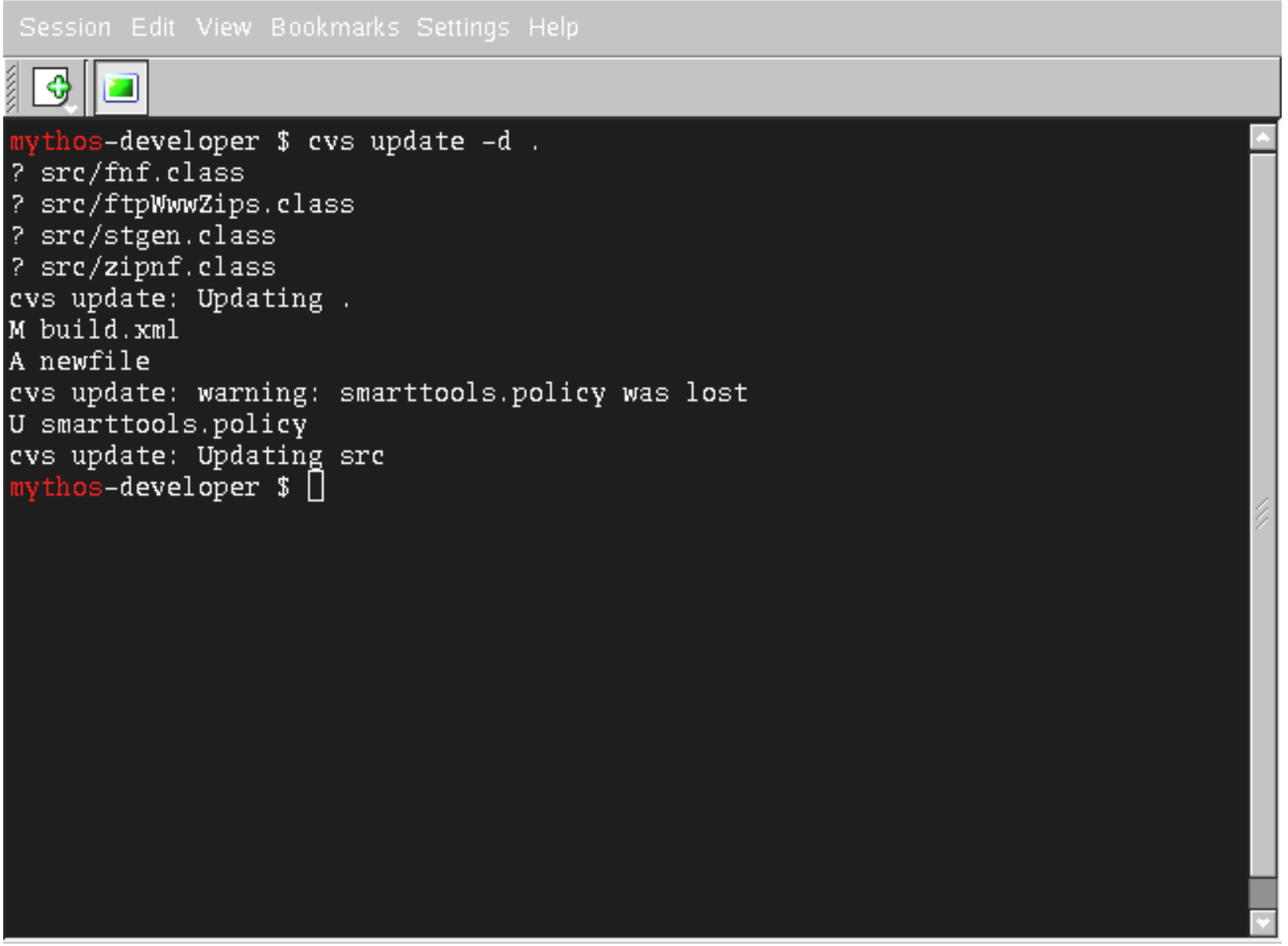
Checkout and update (from the server base into the client directory)



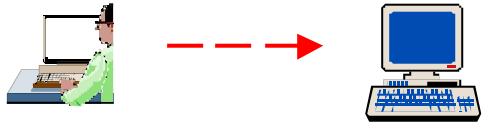
- Checkout:
  - `cvs -d cvsroot checkout module_name`
  - `svn [URL] checkout [PATH]`
- Update:
  - `cvs update –d directories/files`
  - `svn update directories/files`

  - U: updated
  - M(cvs)/G(svn): modified (or merged), the local file is different from the server file and merge was done in the local copy
  - ?(cvs): unknown, this file is present locally but not in the CVS/SVN base
  - A: added, this file is not present locally and has been added
  - R(cvs)/D(svn): removed, this file is present locally and has been removed
  - C: conflict, the local file has been marked with conflict markers which identify the different conflict locations


Session Edit View Bookmarks Settings Help
mythos-test $ cvs -d :ext:drey@cvs-sop.inria.fr:/CVS/SmartTools checkout Documents
cvs checkout: Updating Documents
U Documents/CSSinSmartTools.html
U Documents/Composants.html
U Documents/Composants.txt
U Documents/DidierMemoST
U Documents/Event.txt
U Documents/MissionDream2004.txt
U Documents/SmartTools.ppt
U Documents/compilation.txt
U Documents/packages.html
U Documents/packages.txt
cvs checkout: Updating Documents/PawelKowalski
U Documents/PawelKowalski/changements.txt
U Documents/PawelKowalski/propositions.txt
U Documents/PawelKowalski/rapport.txt
U Documents/PawelKowalski/soapmon.txt
U Documents/PawelKowalski/tests.txt
U Documents/PawelKowalski/todos.txt
cvs checkout: Updating Documents/PawelKowalski/TransformTutorial
U Documents/PawelKowalski/TransformTutorial/TODOS.txt
U Documents/PawelKowalski/TransformTutorial/TransformTutorial.tcp
U Documents/PawelKowalski/TransformTutorial/TransformTutorial.tex
U Documents/PawelKowalski/TransformTutorial/biblio.tex
U Documents/PawelKowalski/TransformTutorial/createCoXSL.tex
U Documents/PawelKowalski/TransformTutorial/generate-component.vsd
U Documents/PawelKowalski/TransformTutorial/generate-wslayer.vsd
U Documents/PawelKowalski/TransformTutorial/generateWsLayer.tex
U Documents/PawelKowalski/TransformTutorial/graphDeDepart.dot
U Documents/PawelKowalski/TransformTutorial/intro.tex
U Documents/PawelKowalski/TransformTutorial/output-dir.dot
U Documents/PawelKowalski/TransformTutorial/overview.vsd
U Documents/PawelKowalski/TransformTutorial/ressource-dir.dot
U Documents/PawelKowalski/TransformTutorial/src-directory.dot
U Documents/PawelKowalski/TransformTutorial/subdivision.tex
U Documents/PawelKowalski/TransformTutorial/tutorial.tex
U Documents/PawelKowalski/TransformTutorial/tutorial.txt


Session Edit View Bookmarks Settings Help
mythos-developer $ cvs update -d .
? src/fnf.class
? src/ftpWwwZips.class
? src/stgen.class
? src/zipnf.class
cvs update: Updating .
M build.xml
A newfile
cvs update: warning: smarttools.policy was lost
U smarttools.policy
cvs update: Updating src
mythos-developer $

Commit (from the client directory into the server base)



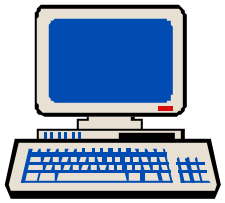
- `cvs/svn commit -m “my message” directories/files`

- Always use explicit messages !!!

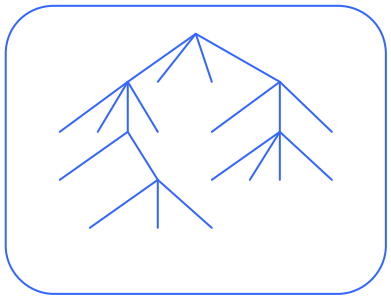
- List of forbidden messages:
  - “ok” → ☺
  - “” → ?
  - “modification of file toto.tex” → which modification ?
  - “bug fixed” → which bug ?
  - …

- Note: with SVN, once a commit is done, all the files have gotten a new revision number

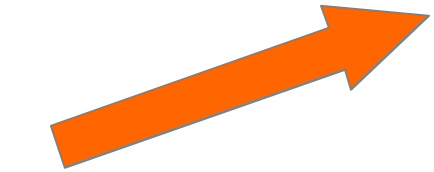
Ideal Development (1/4)

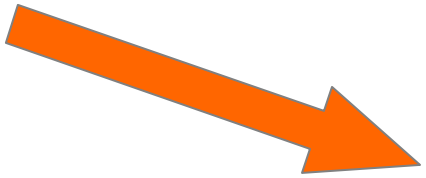


base

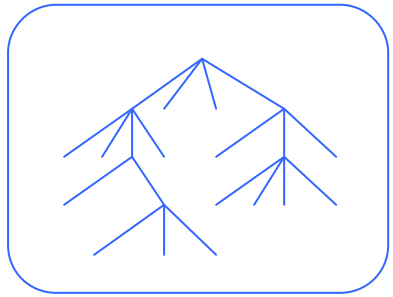


checkout

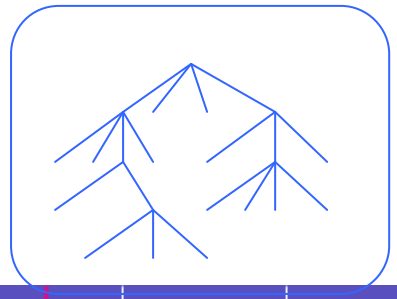




Developer A



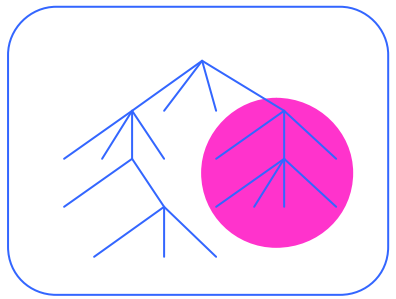
Developer B



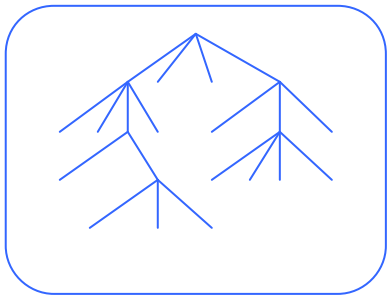
Ideal Development (2/4)

development

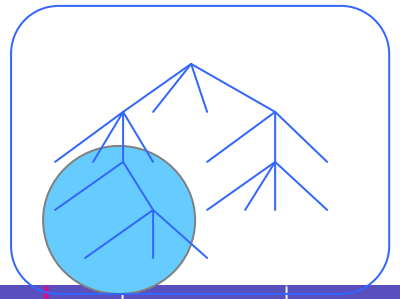
Developer A



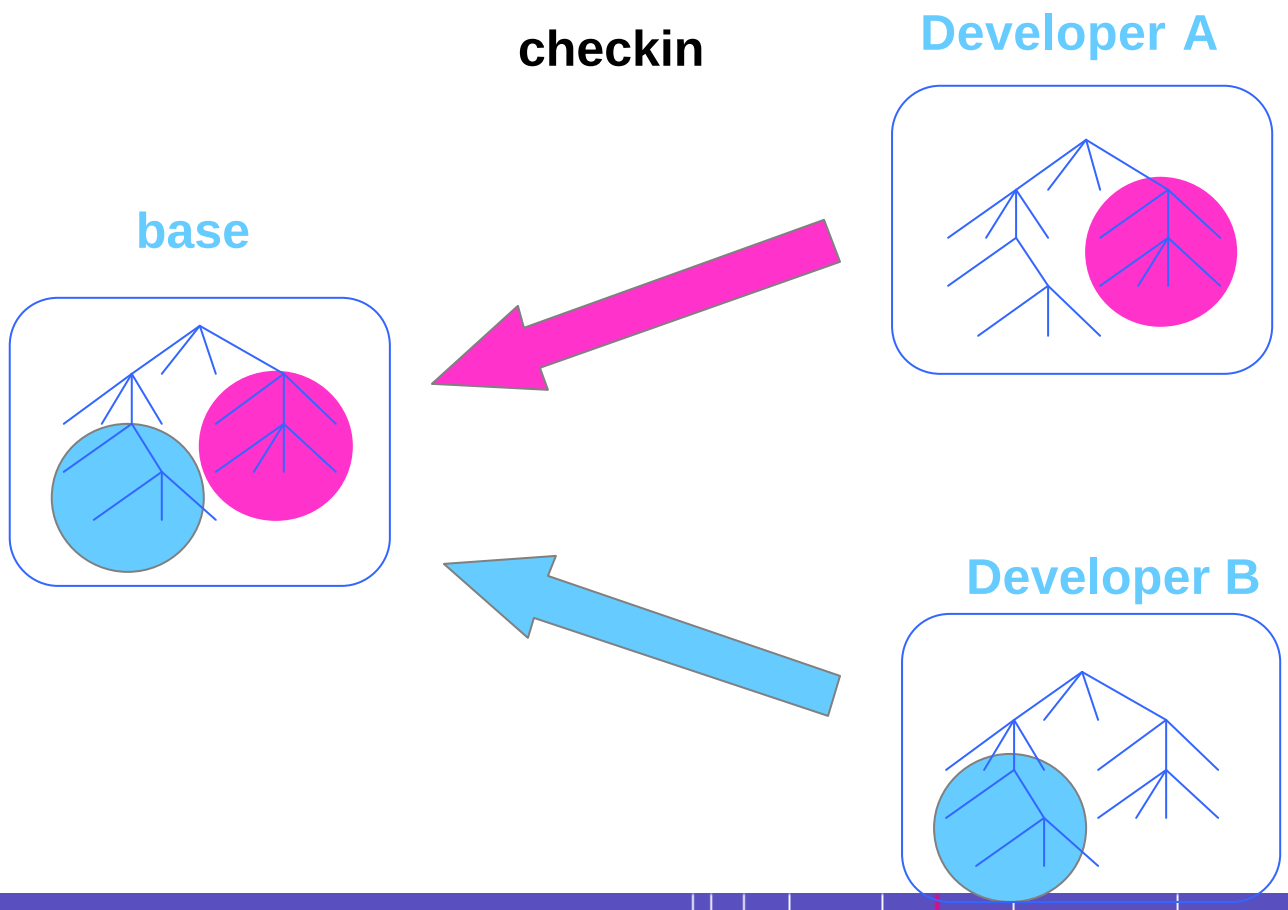
base



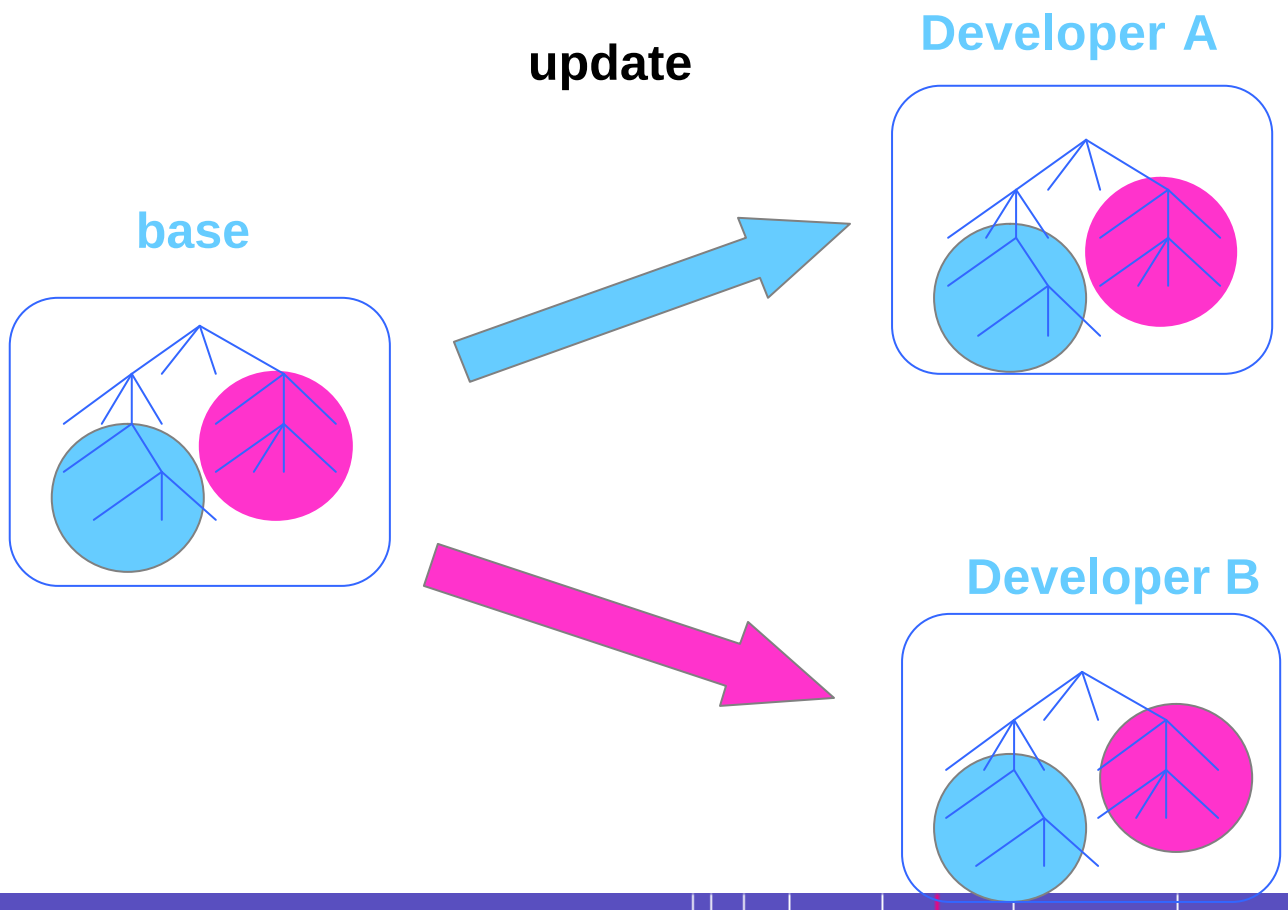
Developer B



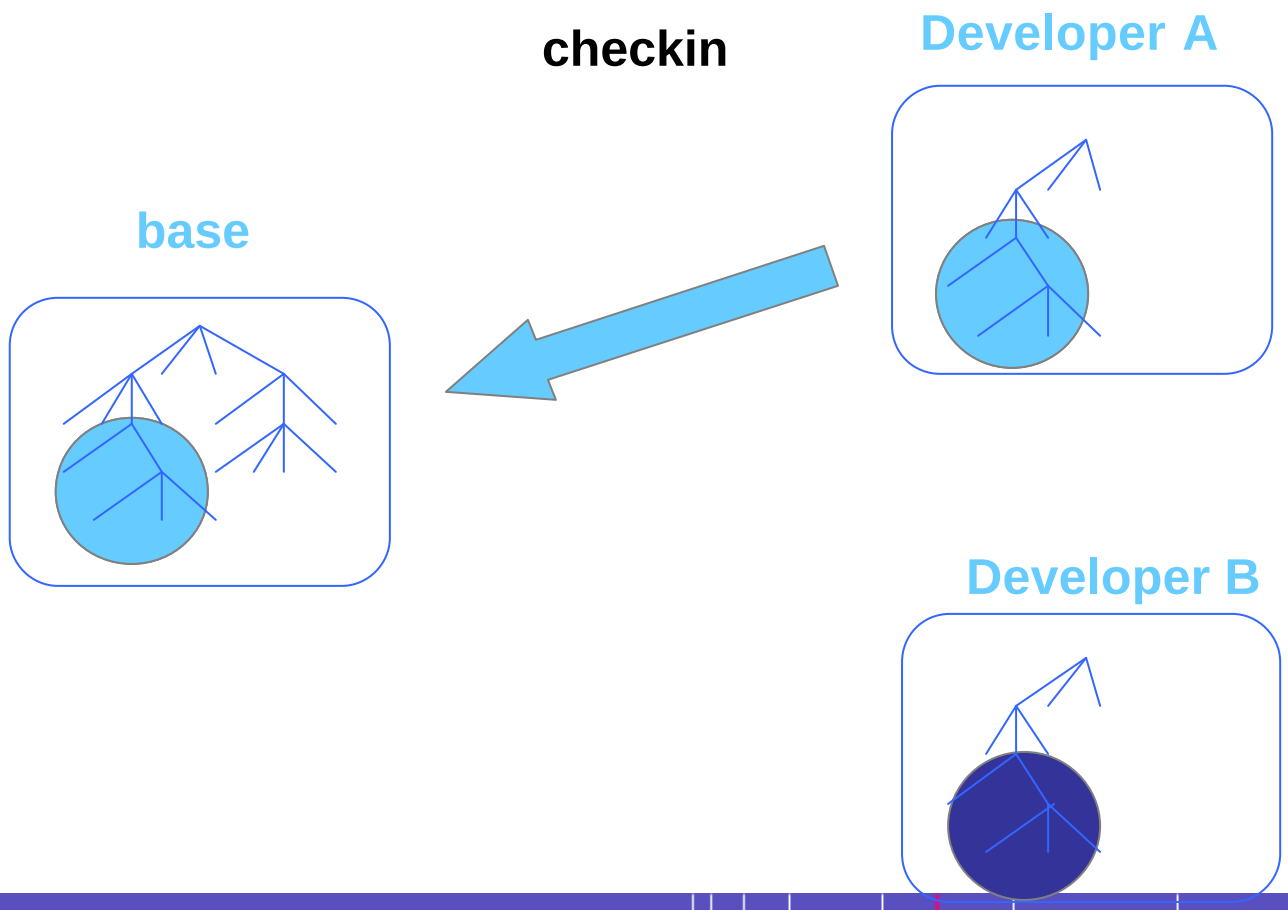
Ideal Development (3/4)



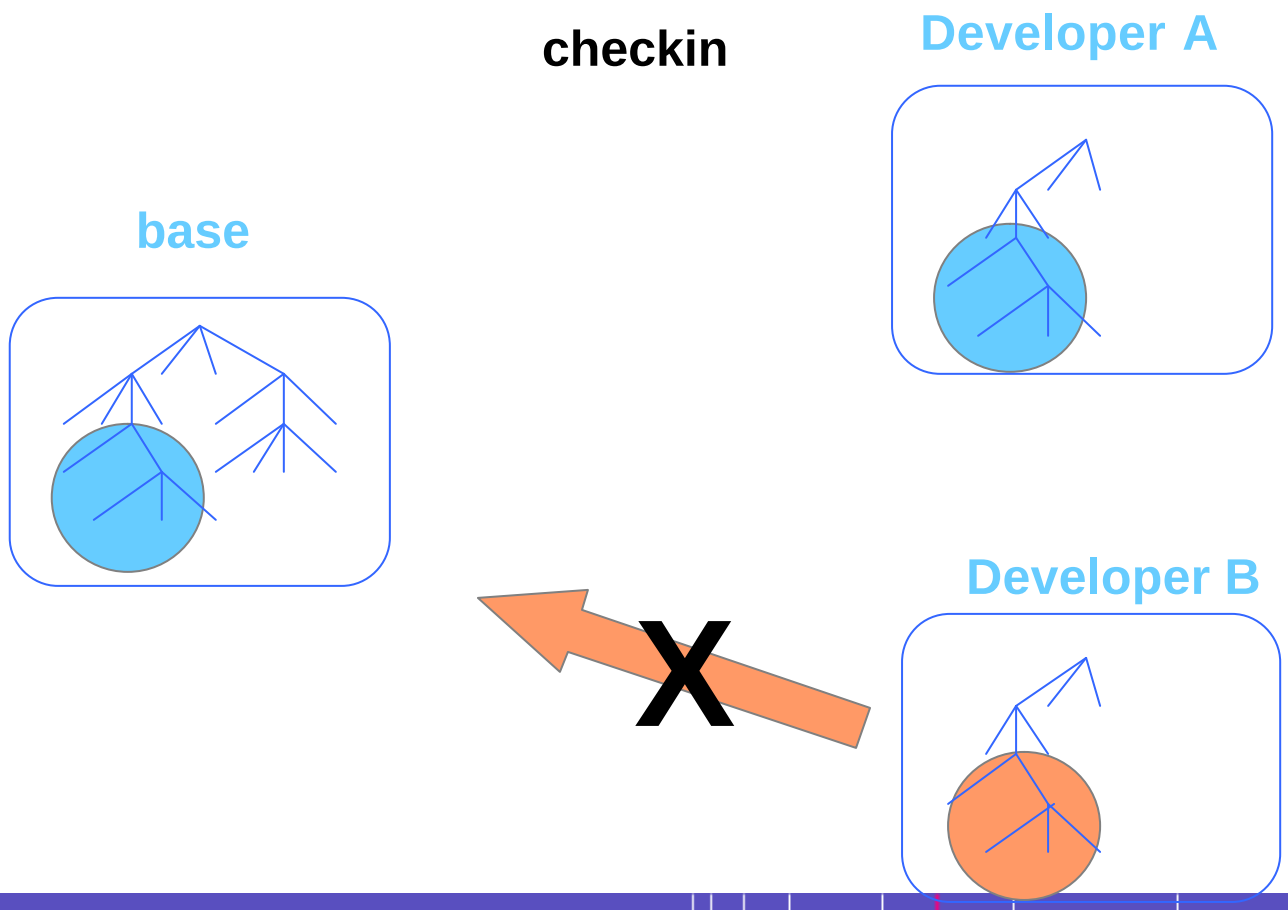
Ideal Development (4/4)



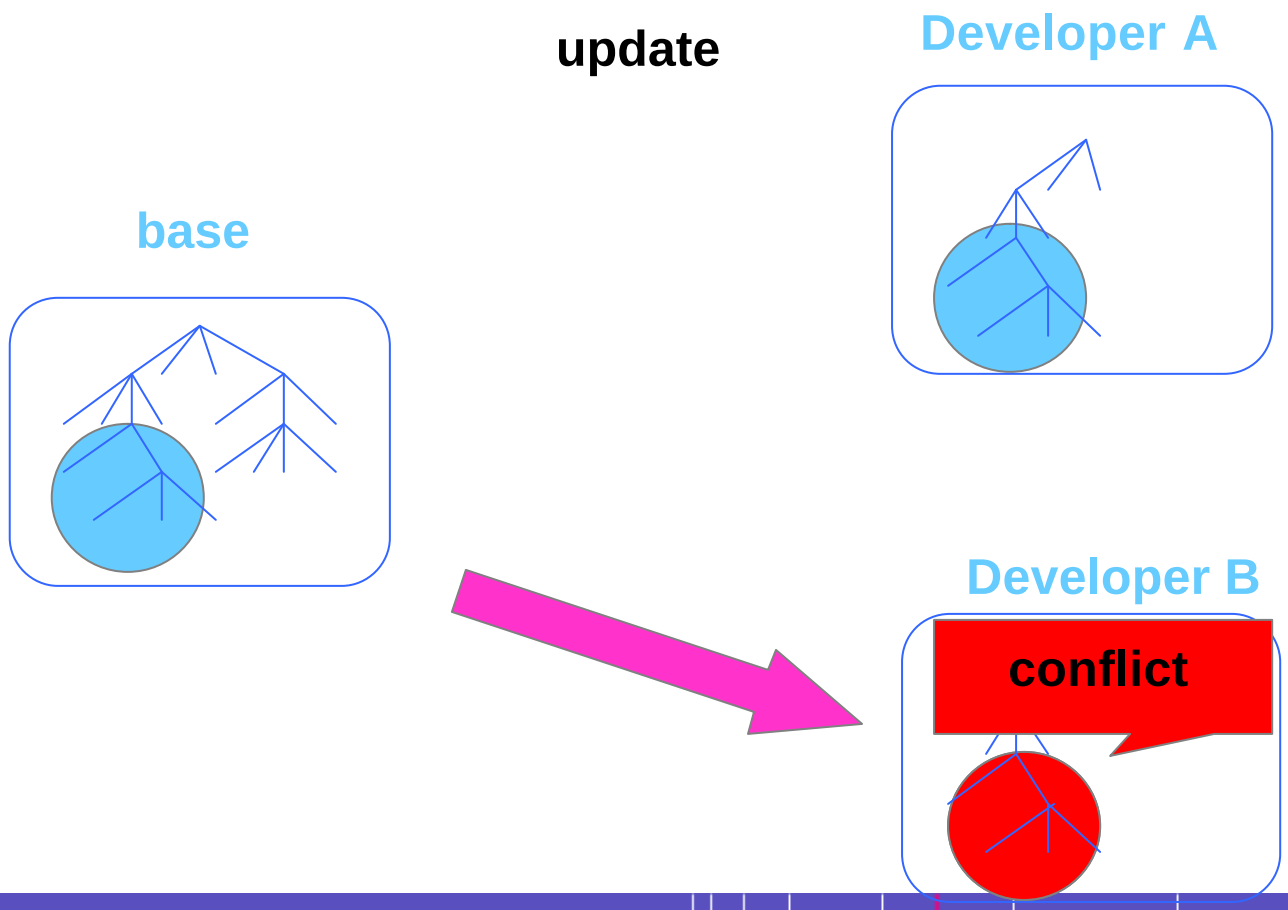
Real Development (1/5)



Real Development (2/5)



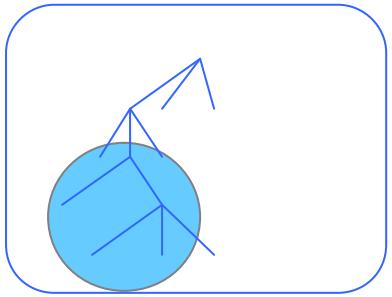
Real Development (3/5)



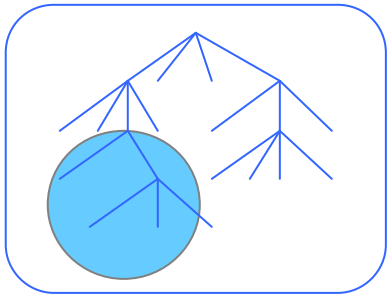
Real Development (4/5)

Conflict Resolution

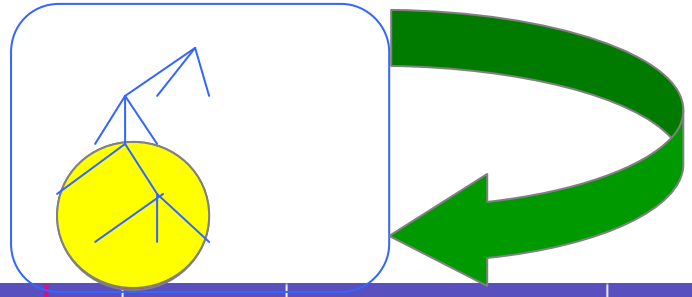
Developer A



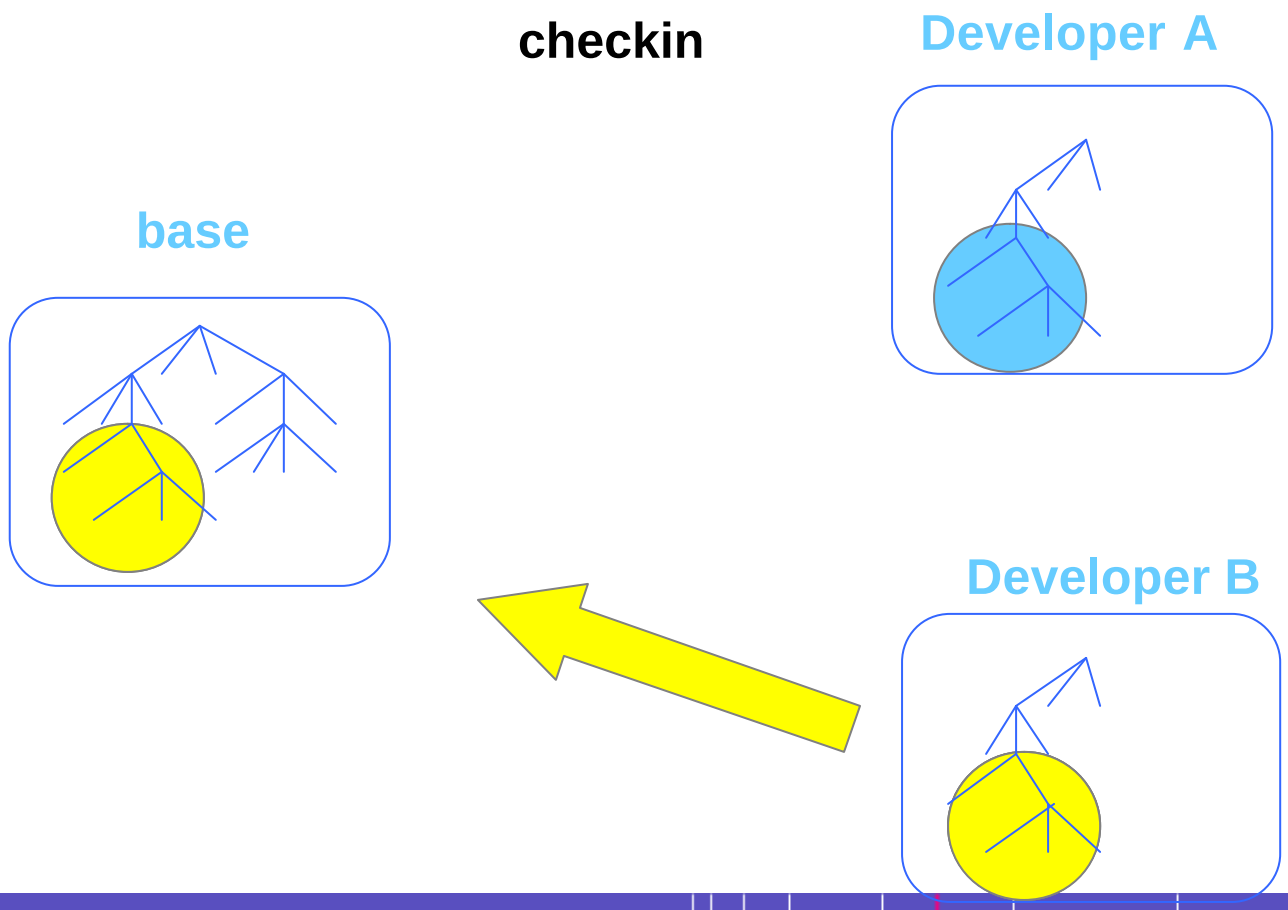
base



Developer B



Real Development (5/5)

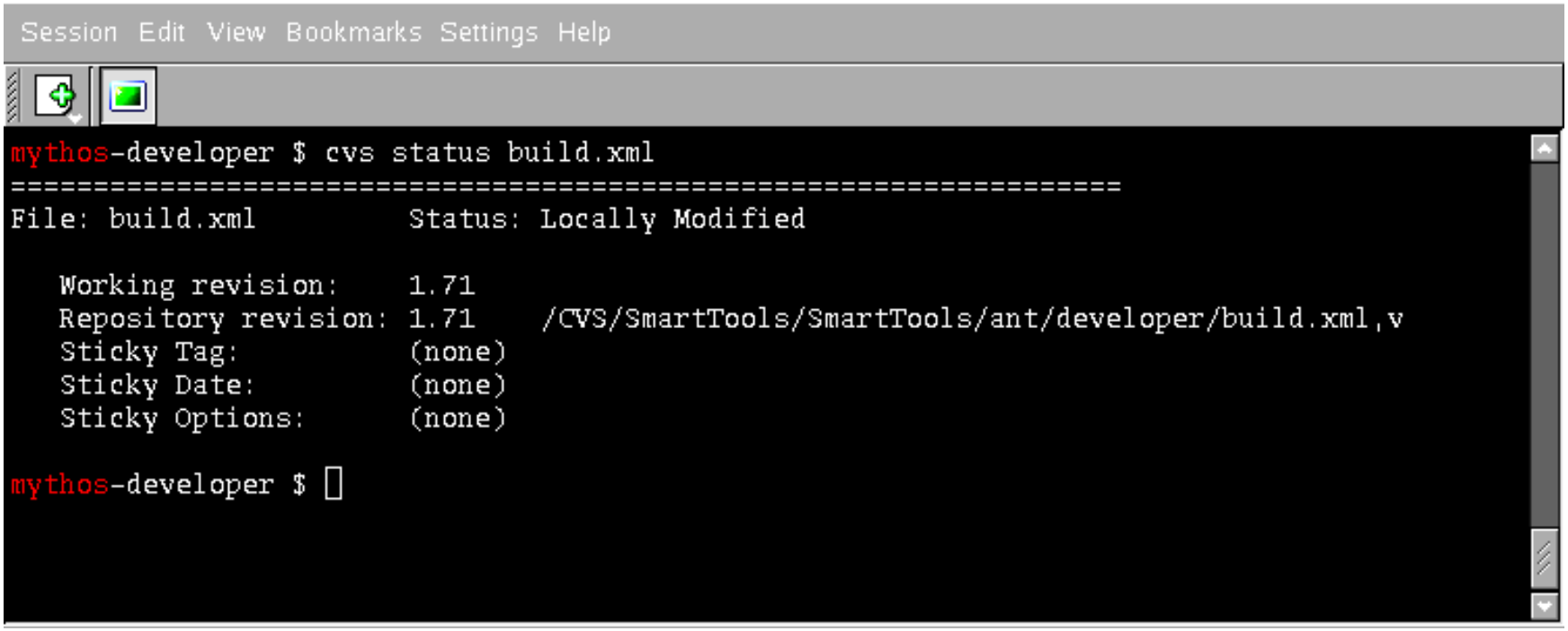


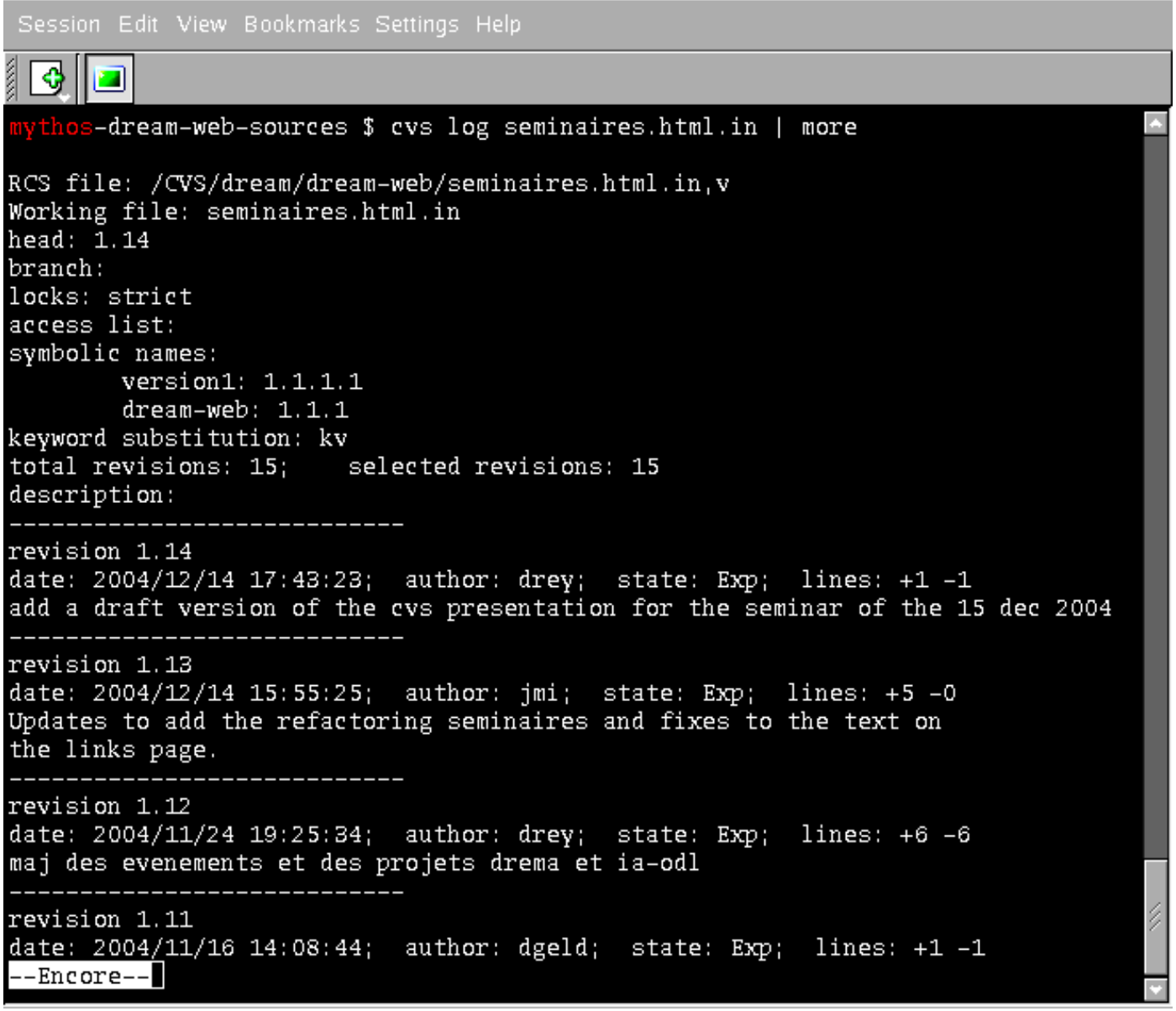
Conflicts

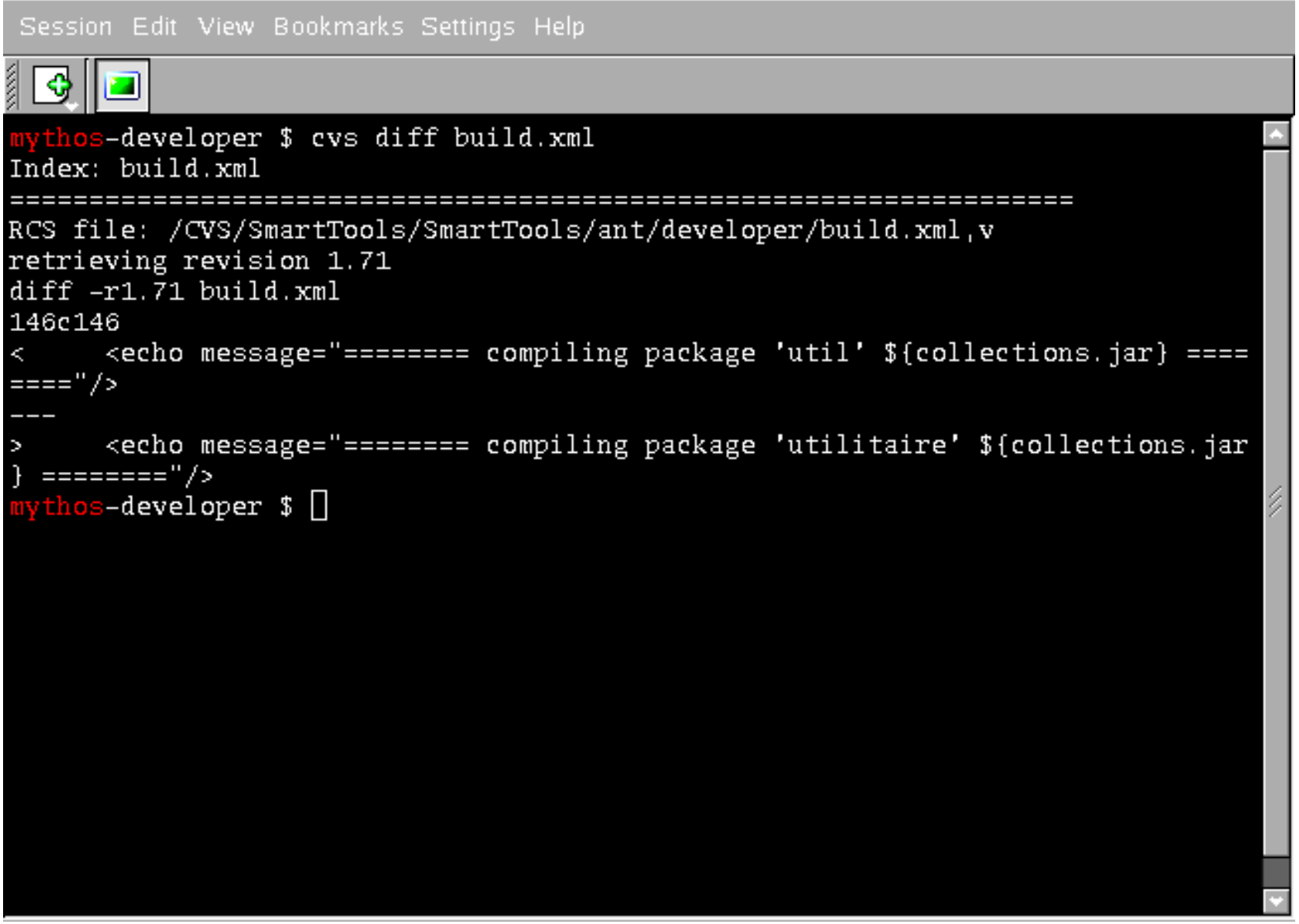
- Do not panic!
  - Many problems are easy to fix
  - Generally, it does not require a lot of interaction
- If necessary, discuss with people to find a compromise
  - Meetings
  - Email
  - Phone
  - ...
- Usually can be avoided with regular updates

Status, log, diff, annotate

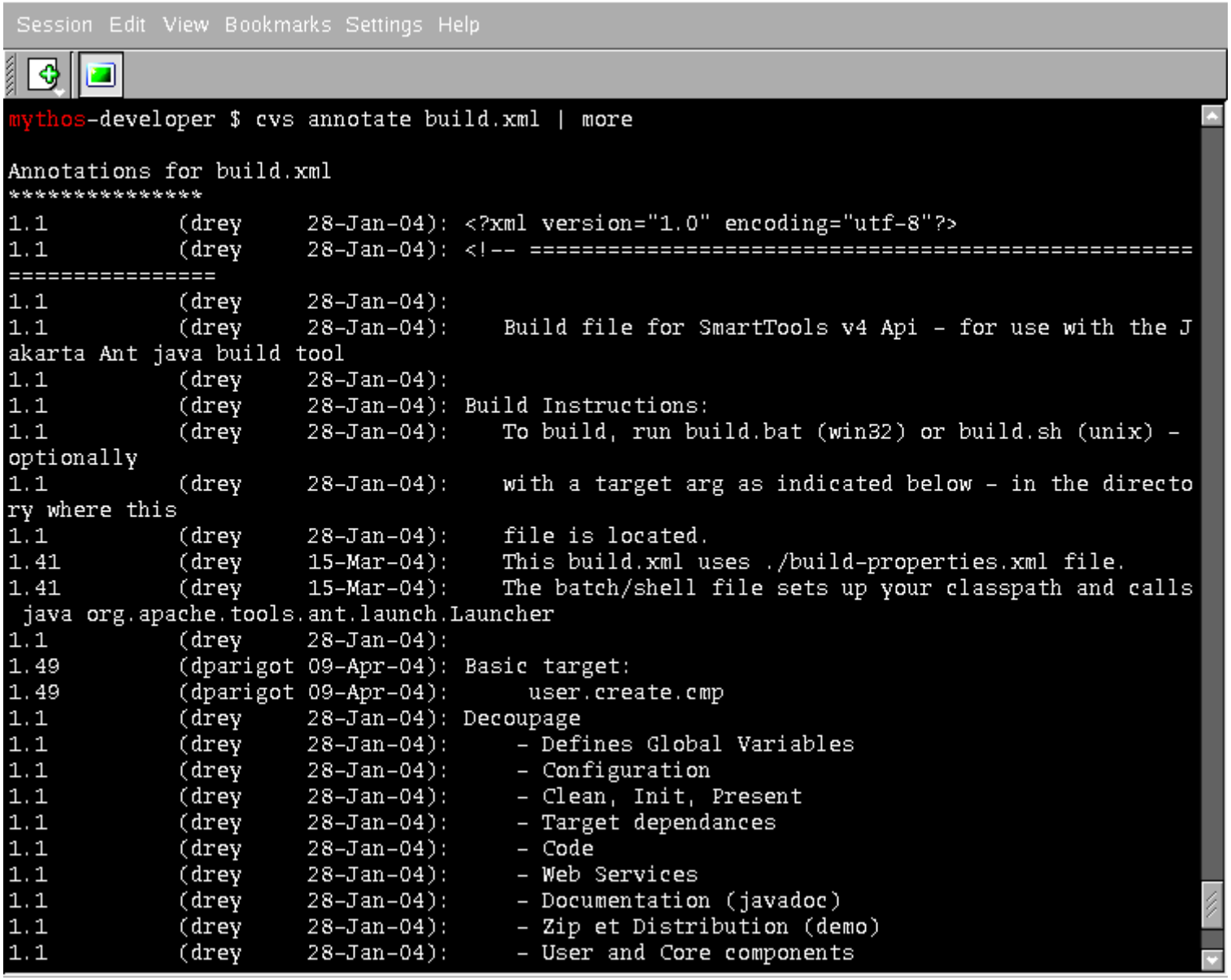
- `cvs/svn status filename`
  - Gives information about local file with comparison to base file
- `cvs/svn log filename`
  - Show all the previous versions numbers, comitter, date, and number of lines modified
- `cvs/svn diff filename`
  - Show lines where there are differences between local and base file
- `cvs/svn annotate filename`
  - Gives information line by line: version of introduction, who, when


Session Edit View Bookmarks Settings Help
mythos-developer $ cvs status build.xml
File: build.xml Status: Locally Modified
Working revision: 1.71
Repository revision: 1.71 /CVS/SmartTools/SmartTools/ant/developer/build.xml,v
Sticky Tag: (none)
Sticky Date: (none)
Sticky Options: (none)
mythos-developer $


Session Edit View Bookmarks Settings Help
mythos-dream-web-sources $ cvs log seminaires.html.in | more
RCS file: /CVS/dream/dream-web/seminaires.html.in,v
Working file: seminaires.html.in
head: 1.14
branch:
locks: strict
access list:
symbolic names:
version1: 1.1.1.1
dream-web: 1.1.1
keyword substitution: kv
total revisions: 15; selected revisions: 15
description:
revision 1.14
date: 2004/12/14 17:43:23; author: drey; state: Exp; lines: +1 -1
add a draft version of the cvs presentation for the seminar of the 15 dec 2004
revision 1.13
date: 2004/12/14 15:55:25; author: jmi; state: Exp; lines: +5 -0
Updates to add the refactoring seminaires and fixes to the text on
the links page.
revision 1.12
date: 2004/11/24 19:25:34; author: drey; state: Exp; lines: +6 -6
maj des evenements et des projets drema et ia-odl
revision 1.11
date: 2004/11/16 14:08:44; author: dgeld; state: Exp; lines: +1 -1
--Encore--


Session Edit View Bookmarks Settings Help
mythos-developer $ cvs diff build.xml
Index: build.xml
===
RCS file: /CVS/SmartTools/SmartTools/ant/developer/build.xml,v
retrieving revision 1.71
diff -r1.71 build.xml
146c146
< <echo message="======== compiling package 'util' ${collections.jar} ====
====="/>

> <echo message="======== compiling package 'utilitaire' ${collections.jar
} ========"/>
mythos-developer $


Session Edit View Bookmarks Settings Help
mythos-developer $ cvs annotate build.xml | more
Annotations for build.xml
****
1.1 (drey 28-Jan-04): <?xml version="1.0" encoding="utf-8"?>
1.1 (drey 28-Jan-04): <!-- ==========
1.1 (drey 28-Jan-04):
1.1 (drey 28-Jan-04): Build file for SmartTools v4 Api - for use with the J
akarta Ant java build tool
1.1 (drey 28-Jan-04):
1.1 (drey 28-Jan-04): Build Instructions:
1.1 (drey 28-Jan-04): To build, run build.bat (win32) or build.sh (unix) -
optionally
1.1 (drey 28-Jan-04): with a target arg as indicated below - in the directo
ry where this
1.1 (drey 28-Jan-04): file is located.
1.41 (drey 15-Mar-04): This build.xml uses ./build-properties.xml file.
1.41 (drey 15-Mar-04): The batch/shell file sets up your classpath and calls
java org.apache.tools.ant.launch.Launcher
1.1 (drey 28-Jan-04):
1.49 (dparigot 09-Apr-04): Basic target:
1.49 (dparigot 09-Apr-04): user.create.cmp
1.1 (drey 28-Jan-04): Decoupage
1.1 (drey 28-Jan-04): - Defines Global Variables
1.1 (drey 28-Jan-04): - Configuration
1.1 (drey 28-Jan-04): - Clean, Init, Present
1.1 (drey 28-Jan-04): - Target dependances
1.1 (drey 28-Jan-04): - Code
1.1 (drey 28-Jan-04): - Web Services
1.1 (drey 28-Jan-04): - Documentation (javadoc)
1.1 (drey 28-Jan-04): - Zip et Distribution (demo)
1.1 (drey 28-Jan-04): - User and Core components

Add, delete, and move files with CVS

- Add: `cvs add filenames`
- Produces a message that explains that a commit is necessary:
- `cvs commit -m "addition of files bla…" directories/files`
- Note: add is not recursive in subfolders

- Files have to be deleted locally first: `rm filenames`
- Remove: `cvs remove filenames`
- Produces a message that explains that a commit is necessary:
- `cvs commit -m "removed files bla…" directories/files`
- Trace of deleted files in Attic directories on the CVS base

- Move = add + delete, user has to explicitely use add and delete, with an explicit log message to keep a trace of the old names

Add, delete, and move folders with CVS

- Folder addition is as simple as file addition; but do not need a commit command to take effect
- Not possible to delete and/or move folders!
  - Except with “hard/dirty” intervention directly on the CVS base (server’s side)

Add, delete, and move files with SVN

- Add: `svn add filenames`
- Produces a message that explains that a commit is necessary:
  - `svn commit -m "addition of files bla…" directories/files`
- Note: addition is recursive in subfolders by default

- Delete: `svn remove filenames`
- Produces a message that explains that a commit is necessary:
  - `cvs commit -m "removed files bla…" directories/files`

- Move: `svn move SRC DEST`

Add, delete, and move folders with SVN

- Exactly the same commands as for files!
- See previous slide ☺

Tagging

- Tag = string marker which is applied to the status of a part of the CVS/SVN hierarchy
- Often used to mark a given version of the cvs repository with a string which refers to a released version of the software
- Easily get back specific versions of the software
  - Use/distribute a given released version
  - Reproduce bugs for a given release version
- Only files are tagged under CVS, everything under SVN
- CVS command:
- `cvs tag –r 1.2 tagname files`
- `cvs rtag tagname files` (latest version in the base is tagged)
- SVN command (using a copy):
- `svn copy [trunk-URL] [tagcp-URL] -m "Tagging the 1.0 release of the ‘myproject’ project."`
- http://svnbook.red-bean.com/en/1.0/ch04s06.html

Summary of the main commands

- cvs [*cvs-options*] *command* [*cmd-options*] [*args*]
- svn *command* [*cmd-options*] [*args*]

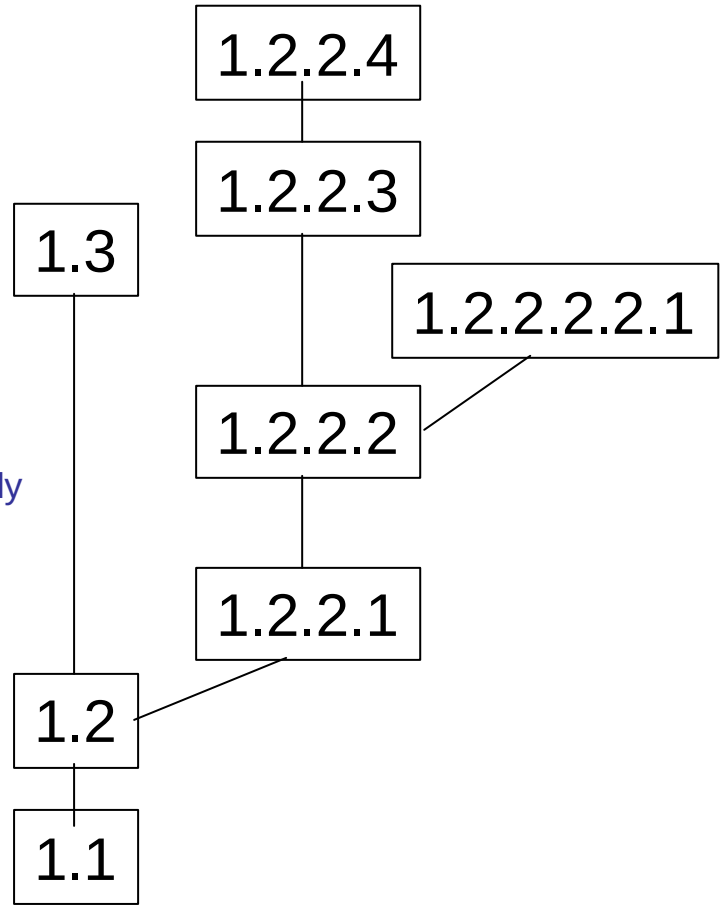
• import	**import files from a local directory to the base**
• checkout	**copy on a local disk a given version of the base**
• commit	**apply the modifications of the local copy to the base**
• update	**upgrade the local copy with a version of the base**
• add	**add a file to the base**
• remove	**remove a file from the base**
• status	**show the status of a local file with respect to the base**
• log	**show the different previous commit stages**
• diff	**show lines in local and server files that differ**
• annotate	**show information line by line of a version in the base**
• tag	**put a tag to identify a given version in the base**

Overview

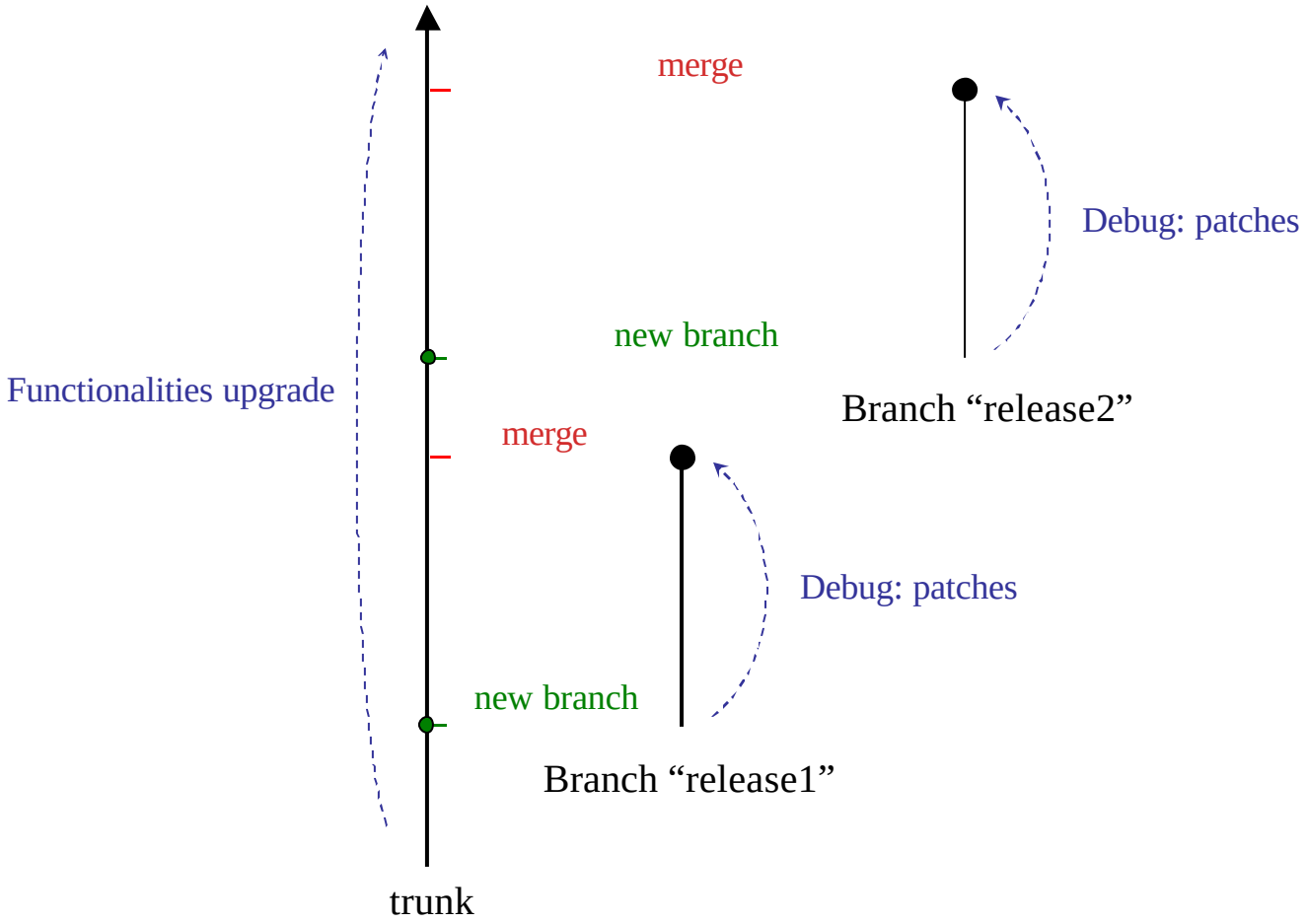
- Collaborative work and version control
- CVS vs. SVN
- Main CVS/SVN user commands
- Advanced use of CVS/SVN

Branching

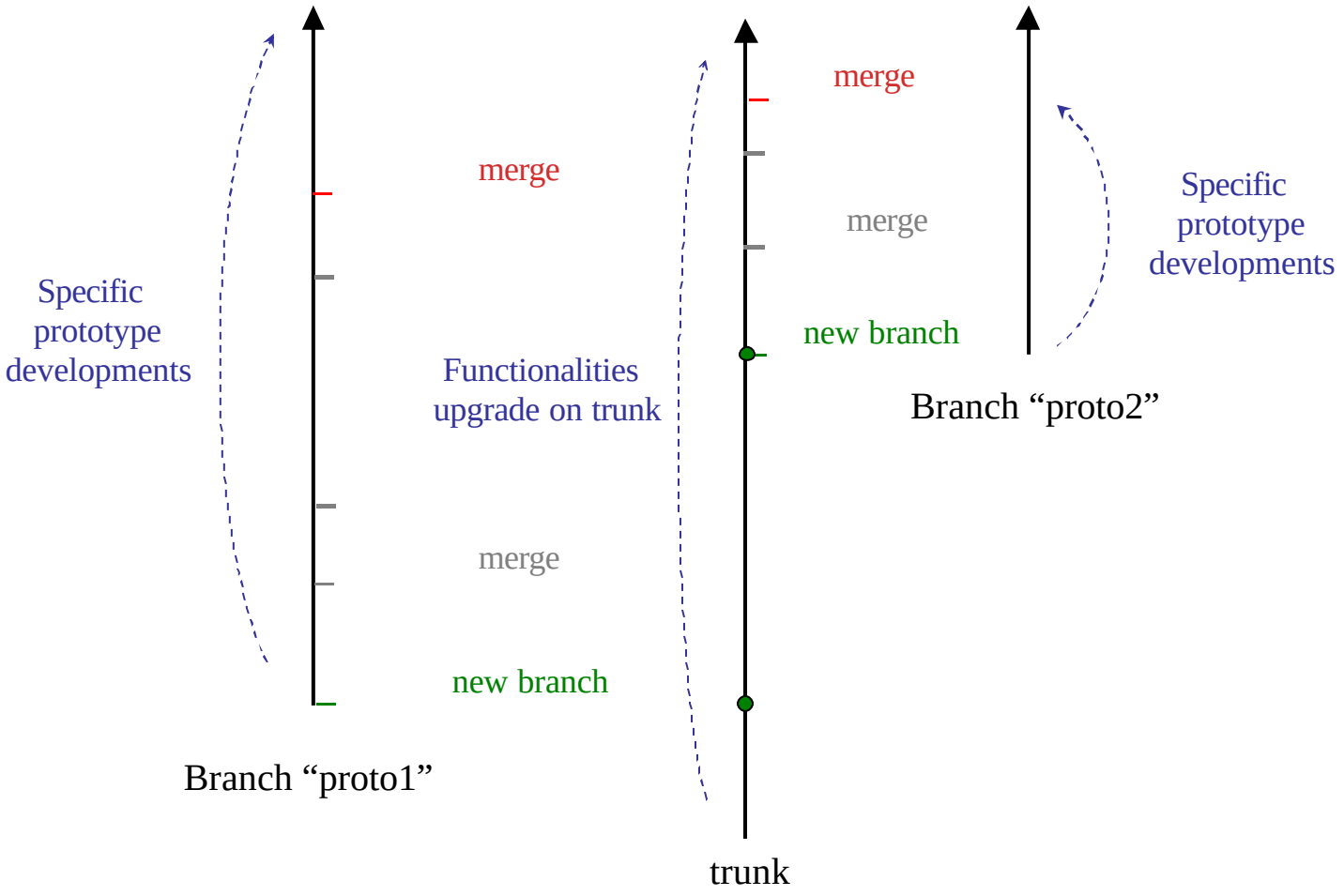
- Branching is easy… merging could be harder!
- When ?
  - Before a release (“freeze” the code in a branch)
  - For parallel development that will not be integrated directly
    - For someone developing during a training period
    - For some “research code”
    - For a specific application (e.g., dedicated to a given client)



Branching: release example



Branching: “prototype” example



Create a branch

- Create a branch in CVS consists in using the *tag* command specifically:
  http://ximbiot.com/cvs/manual/cvs-1.11.18/cvs_5.html
  http://ximbiot.com/cvs/manual/cvs-1.11.18/cvs_5.html#SEC56
  http://www.psc.edu/~semke/cvs_branches.html

- Create a branch with svn consists in copying the trunk:
  http://svnbook.red-bean.com/en/1.0/ch04.html
  http://svnbook.red-bean.com/en/1.0/ch04s02.html#svn-ch-4-sect-2.1
  http://www.cleversafe.org/wiki/Subversion_branching_tutorial

Merge

- Merge with CVS:
  http://ximbiot.com/cvs/manual/cvs-1.11.18/cvs_5.html#SEC60

- Merge with SVN:
  http://svnbook.red-bean.com/en/1.0/re16.html
  http://svnbook.red-bean.com/en/1.0/ch04s04.html

Server administration

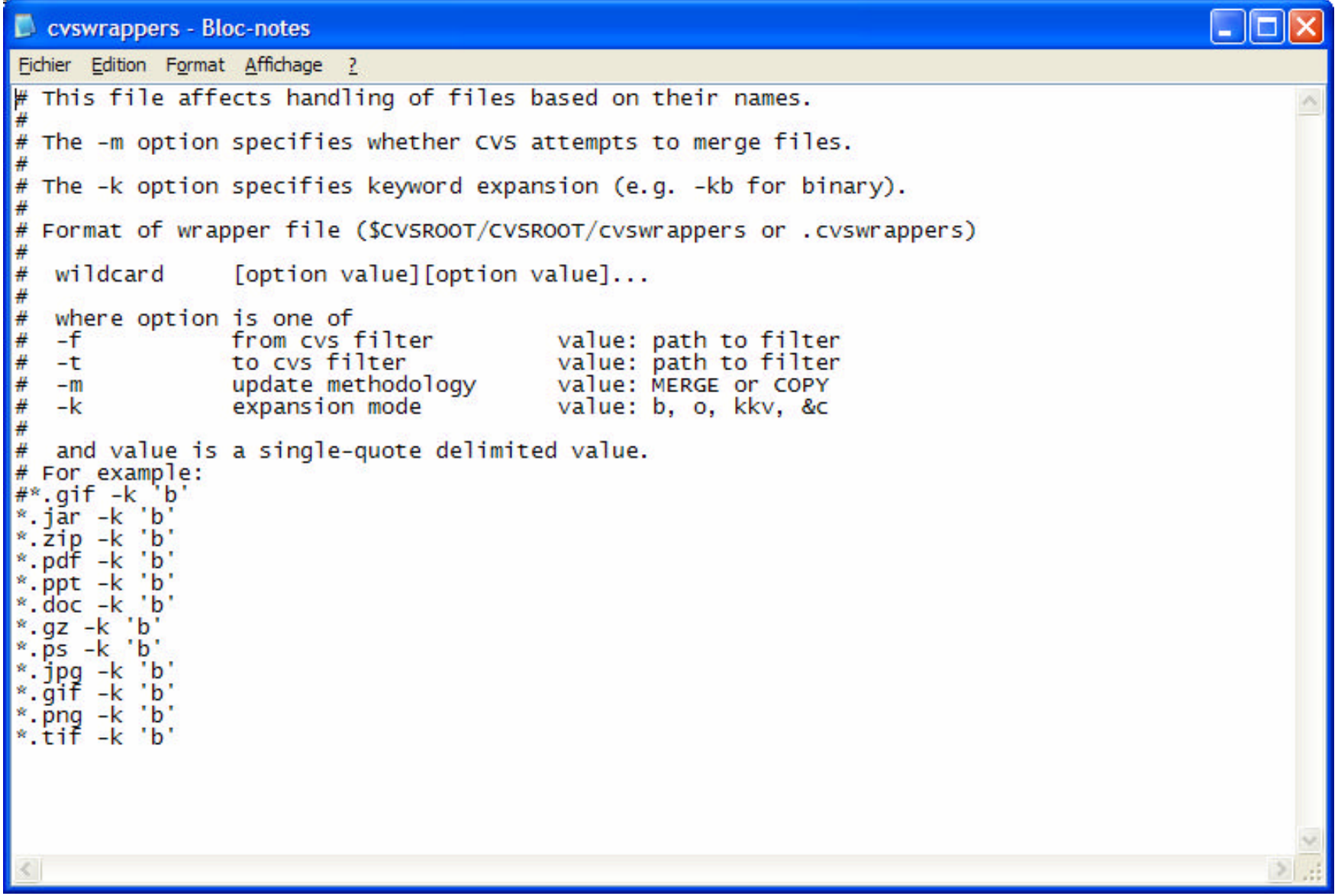
- With CVS:
  - CVSROOT Module (exists by default)
    - cvs checkout CVSROOT
  - Configuration files
    - modules, module definitions
    - cvswrapper binary file control
    - cvsignore list which files CVS has to ignore (*.o, …)
    - commitinfo, editinfo, loginfo, notify, rcsinfo, taginfo make it possible to configure actions with respect to specific CVS operations
  - for folder deletes or similar things…direct manipulations on the server are still possible… 
- With SVN: use *svnadmin* command

Windows users

- Several CVS/SVN clients: webCVS, winCVS, TortoiseCVS, TortoiseSVN, …
- Possible use of CVS/SVN with ssh under Windows via putty (Windows ssh client)
- Problems with line ending (^M under Windows) …
  - **edit files on the platform where the checkout has been done!**

Binary files in CVS base

- No real version control (diffs don’t work)
- Binaries are corrupted if treated as ASCII files
- Can be done with command line: `cvs add –kb filename`
- Can be forced for known extensions
  - Configuration file: cvswrapper


cvswrappers - Bloc-notes
Fichier Edition Format Affichage ?
This file affects handling of files based on their names.
#
The -m option specifies whether CVS attempts to merge files.
#
The -k option specifies keyword expansion (e.g. -kb for binary).
#
Format of wrapper file ($CVSROOT/CVSROOT/cvswrappers or .cvswrappers)
#
wildcard   [option value][option value]...
#
where option is one of
-f   from cvs filter   value: path to filter
-t   to cvs filter   value: path to filter
-m   update methodology   value: MERGE or COPY
-k   expansion mode   value: b, o, kkv, &c
#
and value is a single-quote delimited value.
For example:
#*.gif -k 'b'
*.jar -k 'b'
*.zip -k 'b'
*.pdf -k 'b'
*.ppt -k 'b'
*.doc -k 'b'
*.gz -k 'b'
*.ps -k 'b'
*.jpg -k 'b'
*.gif -k 'b'
*.png -k 'b'
*.tif -k 'b'

Recursion with CVS

- Check each command to see if it is recursive or not
  - Add is not recursive, update is recursive, …
- Read the cvs manual (note we are in the « advanced section » of the expose)

Links

- CVS:
  - http://ximbiot.com/cvs/
  - http://ximbiot.com/cvs/manual/
- SVN:
  - http://subversion.tigris.org/
  - http://artis.imag.fr/Membres/Xavier.Decoret/resources/svn/
  - http://svnbook.red-bean.com/
  - cvs2svn: http://cvs2svn.tigris.org/
- GUI:
  - http://www.twobarleycorns.net/tkcvs.html
  - http://www.wincvs.org
  - http://www.tortoisecvs.org
  - http://tortoisesvn.tigris.org
  - http://www.jalindi.com/igloo
  - http://www.jcvs.org

Closing remarks & Questions

- Use version control systems **from the start**
  - When working alone or in a group
  - For development projects or scientific papers
  - Especially in the case of you need released versions of the software
- Handy FAQ's on the CVS / SVN / INRIA-gforge home sites