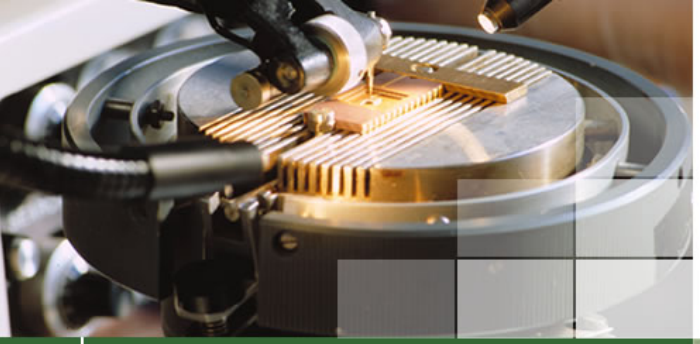


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Planning and Managing Software Projects 2012-13

Session 13

Version Control Systems

Emanuele Della Valle, Lecturer: Daniele Dell'Aglio

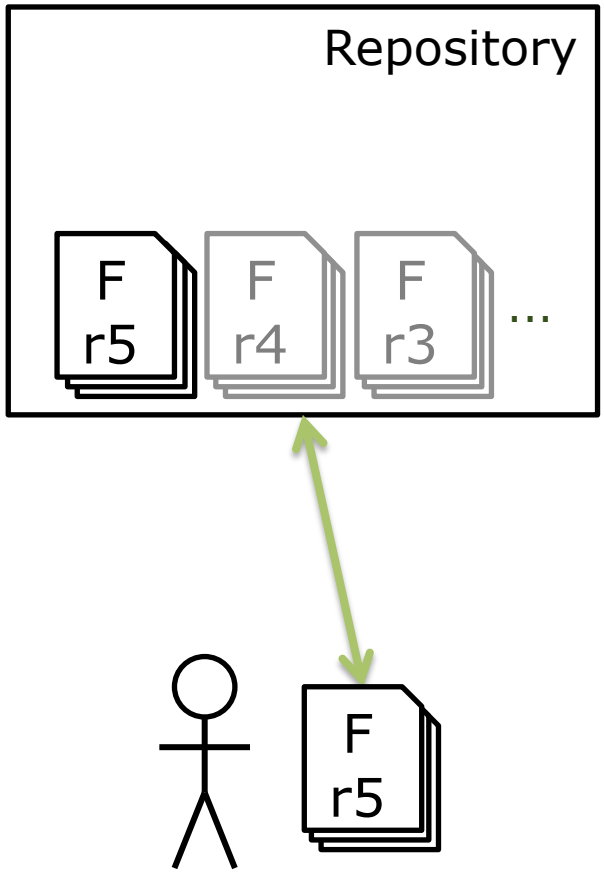
http://dellaglio.org

Outline

- Why version control
- Main concepts
- Basic operations
  - Checkout
  - Commit
  - Update
- Branches and tags

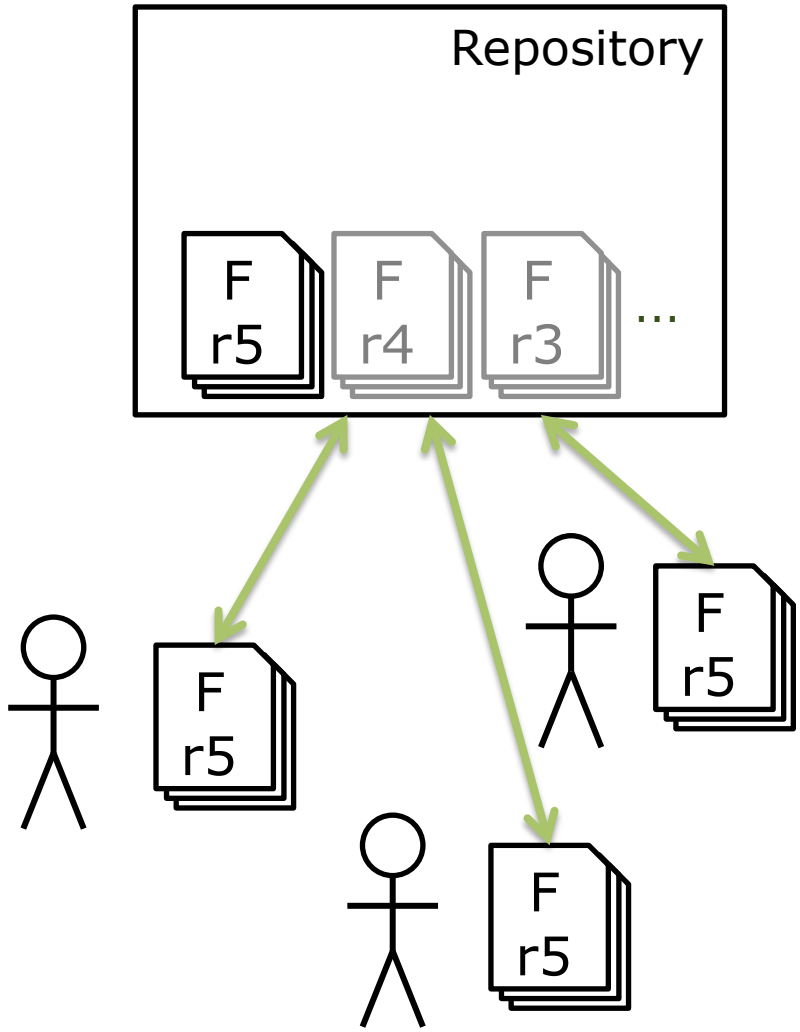
Motivations

- When working alone
  - Backup copy
  - We don’t want to lose the history of our code/documents/ etc.



Motivations

- When working alone
  - Backup copy
  - We don’t want to lose the history of our code/documents/ etc.
- When working in team
  - Share modifications easily
  - Support for cooperative code development



Motivations

- Write code, realize that it is wrong and go back
- Have backups
- Maintain multiple versions of the software
- Compare and find differences between two (or more) versions of your code
- Prove that a particular change broke or fixed some piece of code
- Check how much work is being done (where/when/who)?
- Experiment with new features without interfering with working code

[from: http://stackoverflow.com/questions/1408450/why-should-i-use-version-control]

Verision control system rules (1)

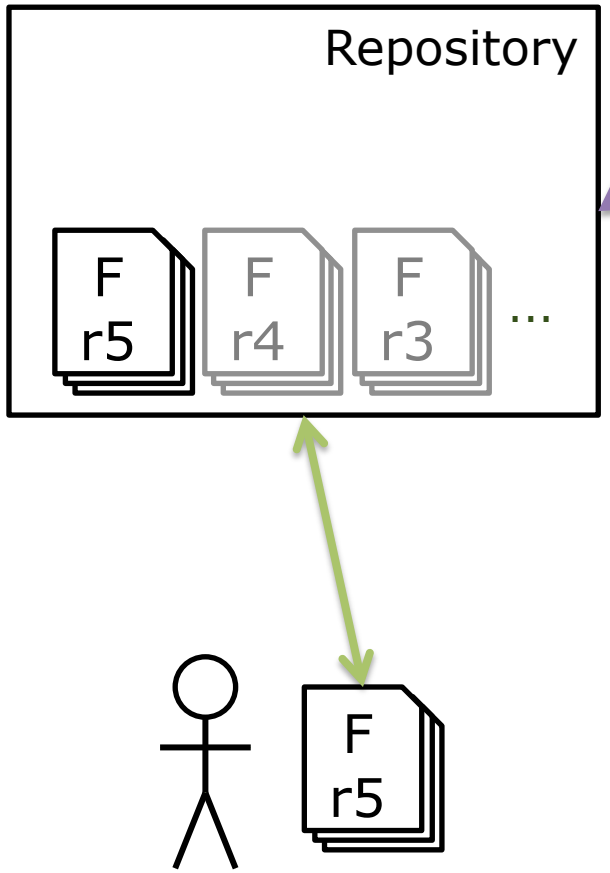
1. If it is not under version control, it doesn't exist

- Do not use e-mail, Skype, Pastebin, Facebook, etc. to exchange the code
- Do not think that it is useless to maintain old versions of the code
- Do not think that code changes can be tracked through paper/chat logs/e-mail

SVN

- There are several Version Control Systems (both open and closed source)
  - CVS
  - SVN
  - Perforce
- In the following we will consider SVN
  - The main concept are similar in the other systems
- Several GUIs for SVN are available
  - TortoiseSVN (Windows)
  - RabbitVCS (Linux)
  - Subclipse/Subversive (Eclipse plug-ins)
  - …

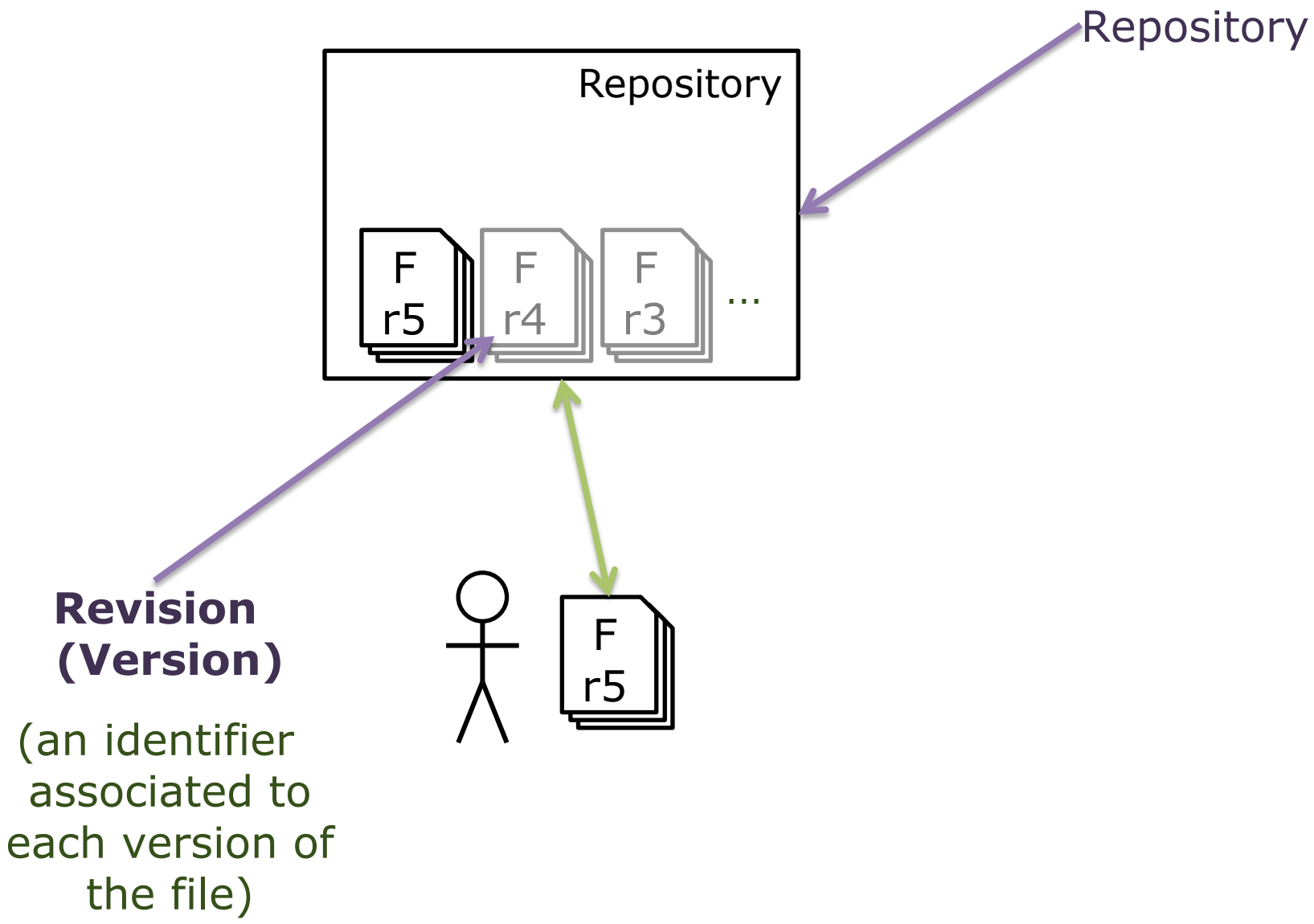
Some definitions



Repository

(the location where files are stored)

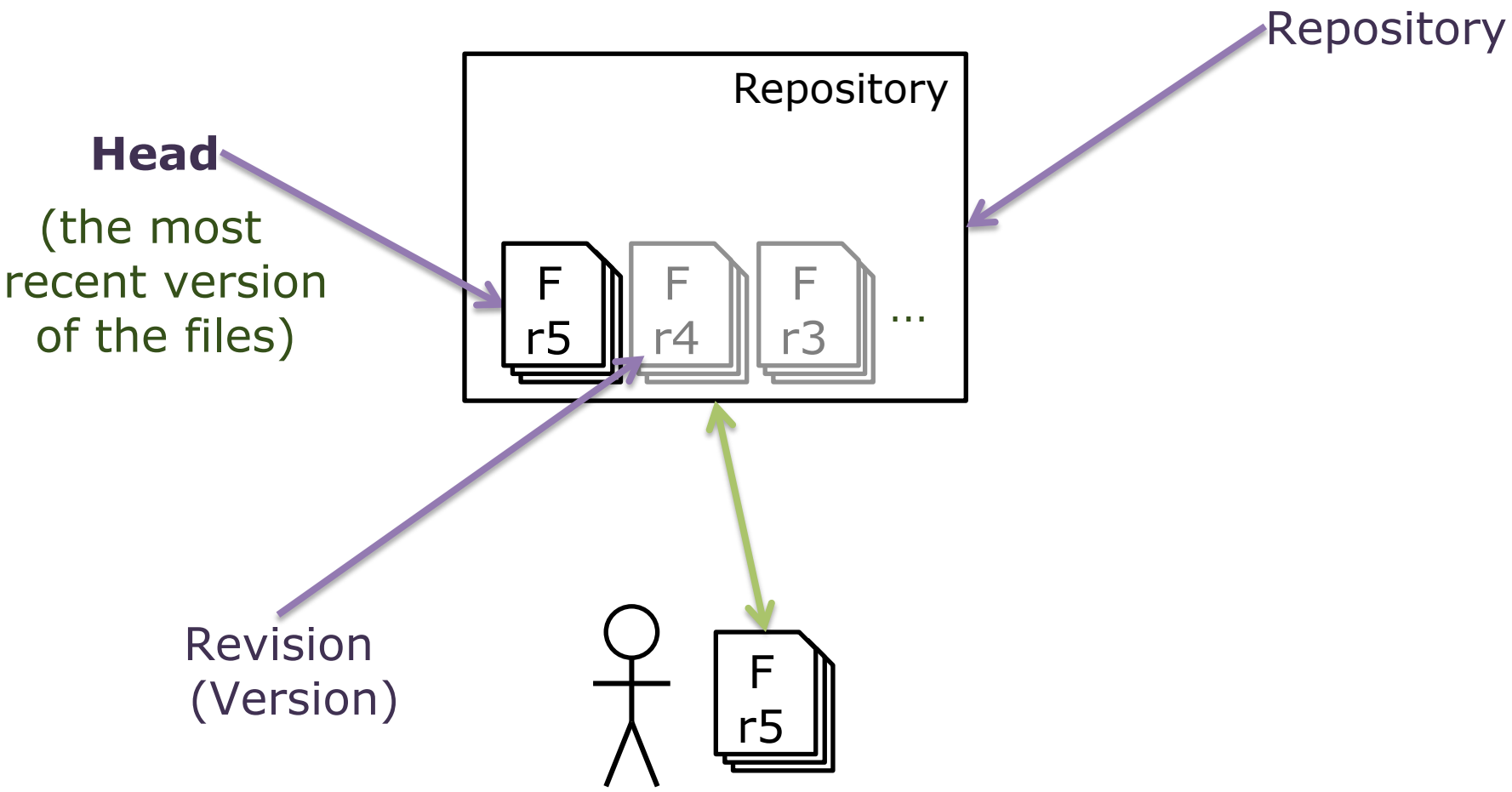
Some definitions



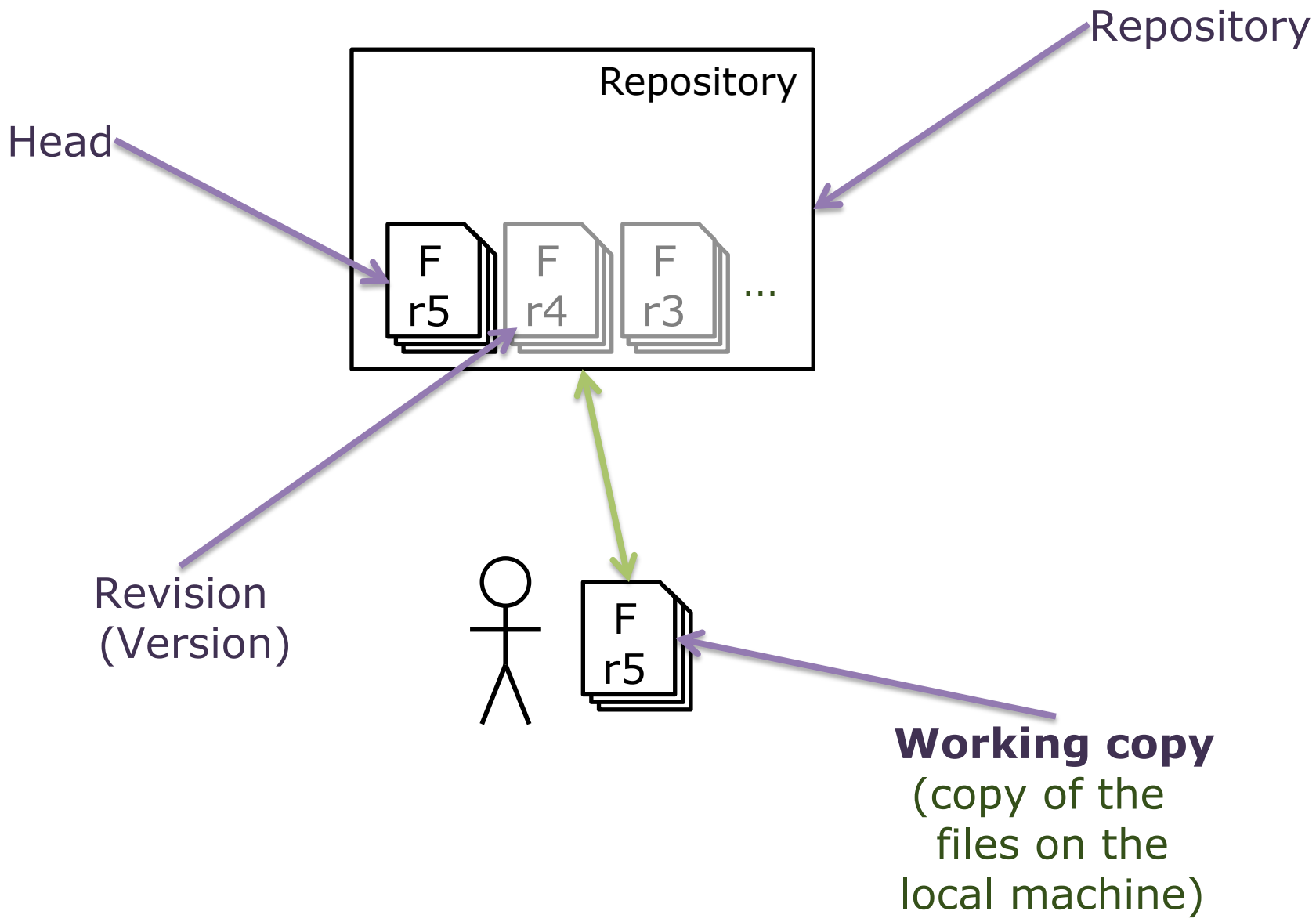
Revision (Version)

(an identifier associated to each version of the file)

Some definitions

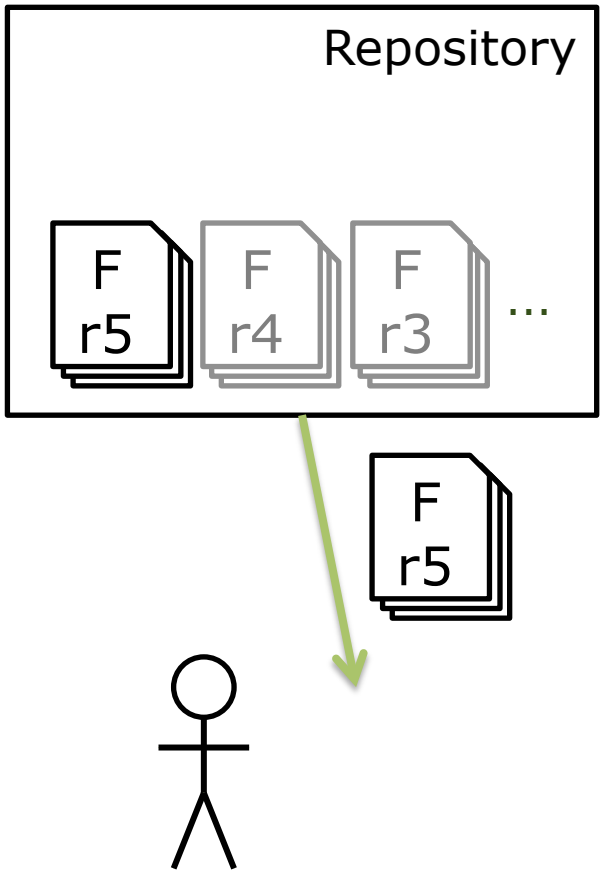


Some definitions



Main operations

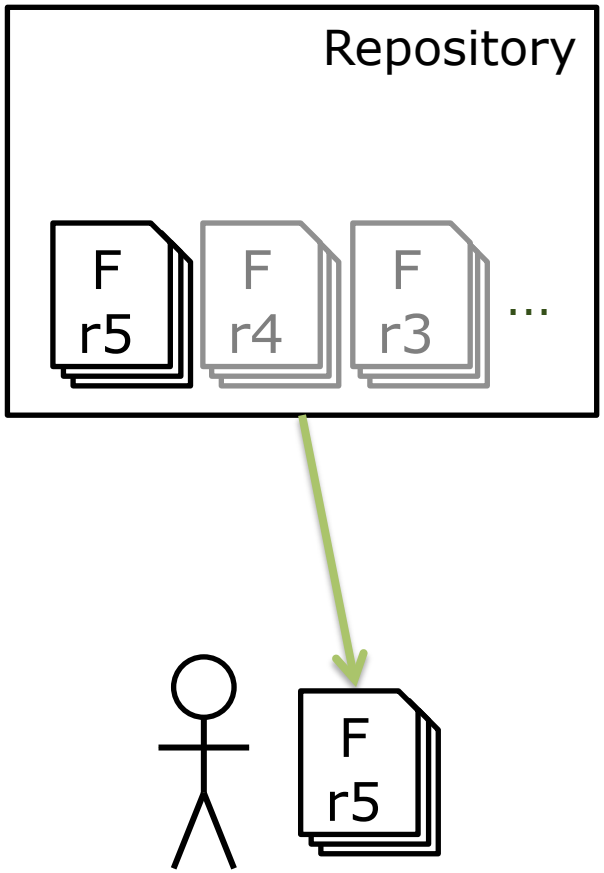
- **Check-out**: initial creation of the working copy from the repository




```
svn checkout <URL>
```

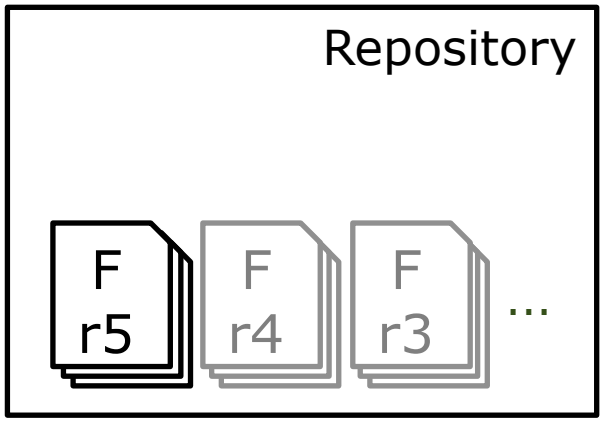
Main operations

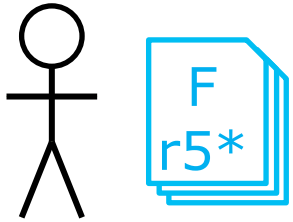
- **Check-out**: initial creation of the working copy from the repository



Main operations

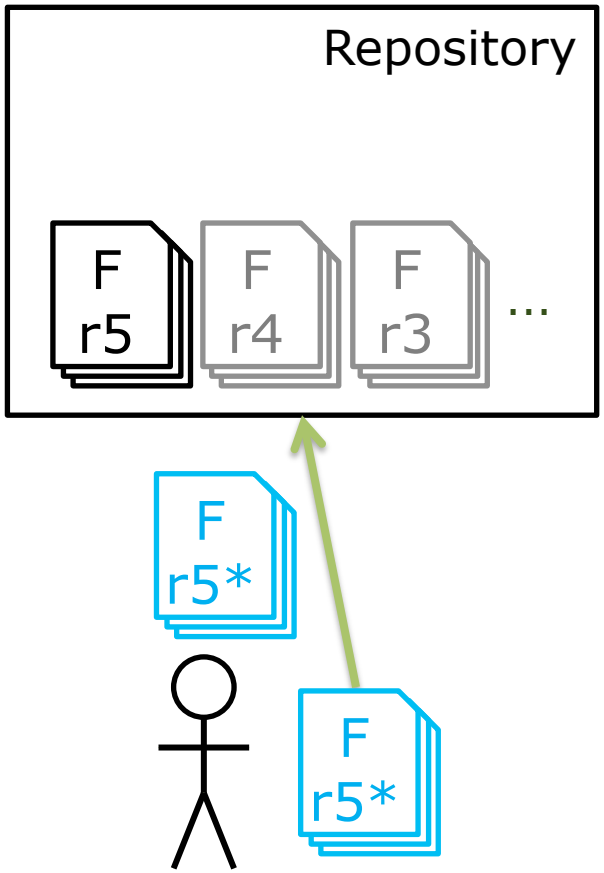
- **Check-out**: initial creation of the working copy from the repository





Main operations

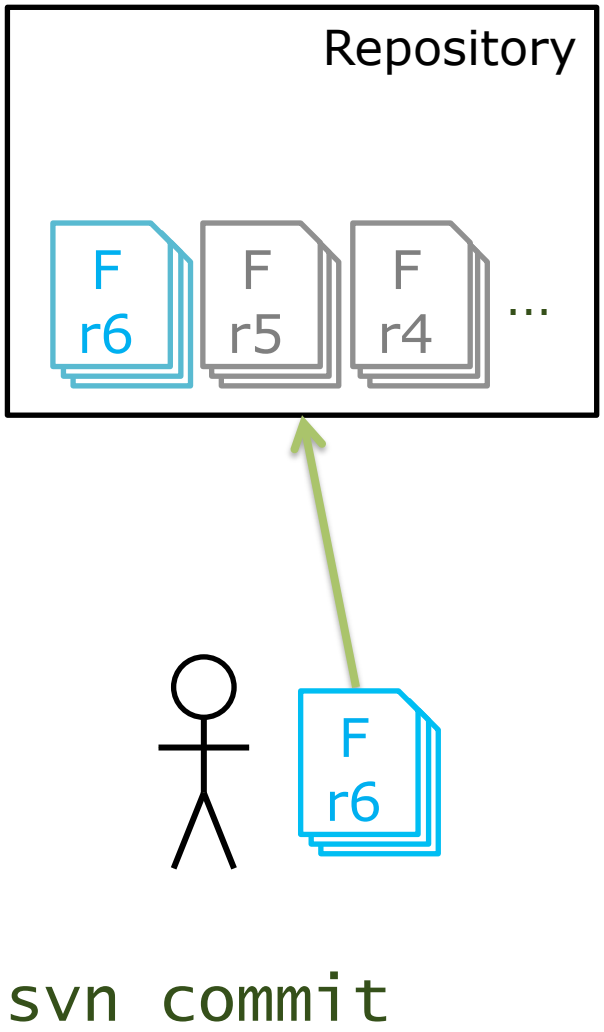
- Check-out: initial creation of the working copy from the repository
- **Check-in** (Commit): update of the HEAD revision with the working copy



`svn commit`

Main operations

- Check-out: initial creation of the working copy from the repository
- **Check-in** (Commit): update of the HEAD revision with the working copy



The commit message

- When you commit a file, you can add a message

```
svn commit -m "message"
```

- Explain why you are making a commit
  - Did you fix bugs? If yes, which ones?
  - Did you add new features? If yes, which ones?
- It is useful for everyone
  - The list of improvements in the code can be easily retrieved
  - Everyone knows what the team is doing
- Do not copy your previous message
  - Some SVN clients maintain a message history
  - You cannot fix two times the same bug…

Version Control System rules (2)

2. Remember the axe-murderer when writing commit messages:

Write every commit message like the next person who reads it is an axe-wielding maniac who knows where you live

(anonymous)

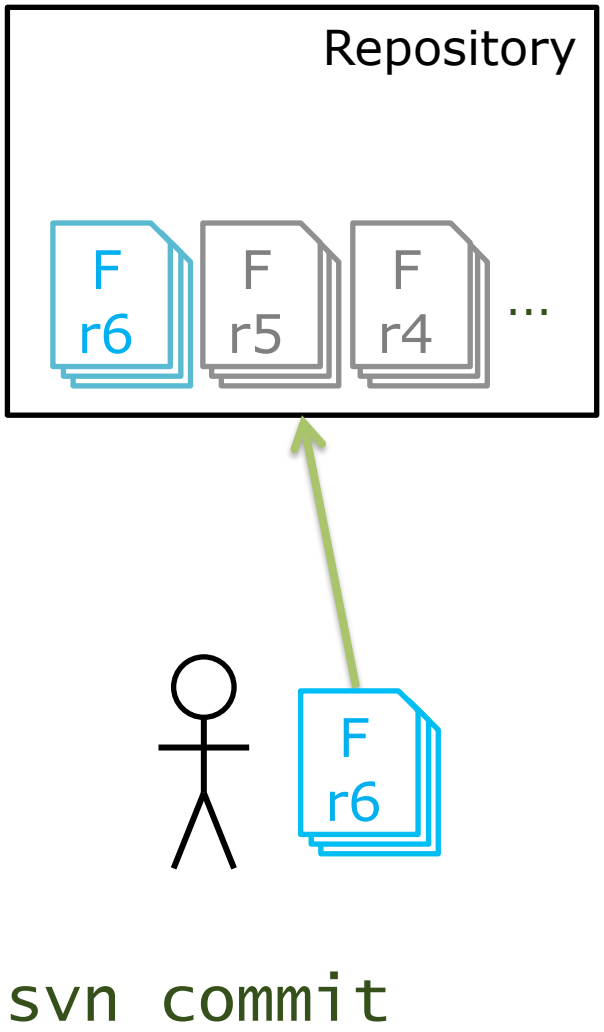
- Examples of bad commit messages:

`It works!`	`Fixed a little bug...`
`Now it works!`	`typo`
`Updated`	`Revision 1024!!`

- Example of a good commit message:
  - http://websvn.kde.org/?view=revision&revision=1355161
- Take a look here:
  - http://techbase.kde.org/Policies/SVN_Commit_Policy#Special_keywords_in_SVN_log_messages

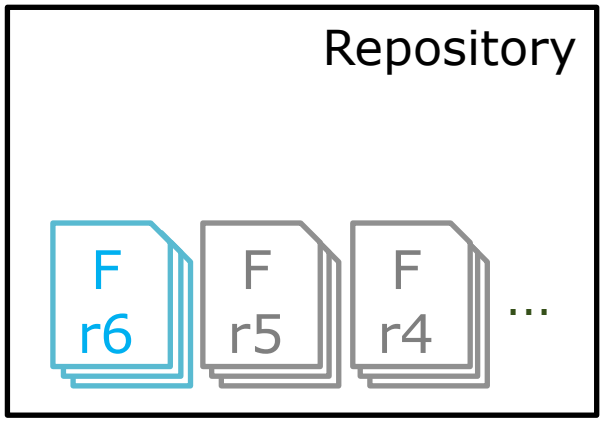
Main operations

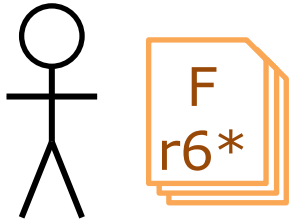
- Check-out: initial creation of the working copy from the repository
- **Check-in** (Commit): update of the HEAD revision with the working copy



Main operations

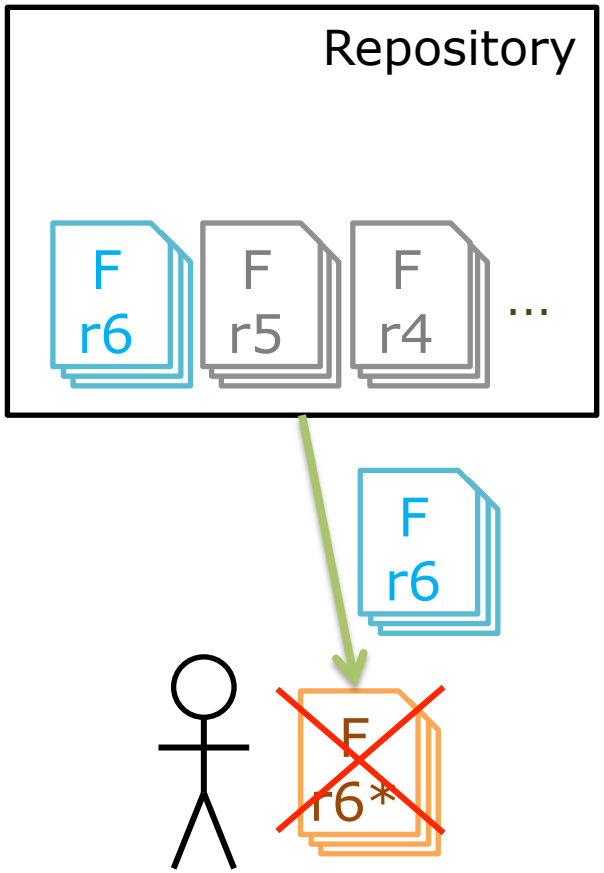
- Check-out: initial creation of the working copy from the repository
- **Check-in** (Commit): update of the HEAD revision with the working copy





Main operations

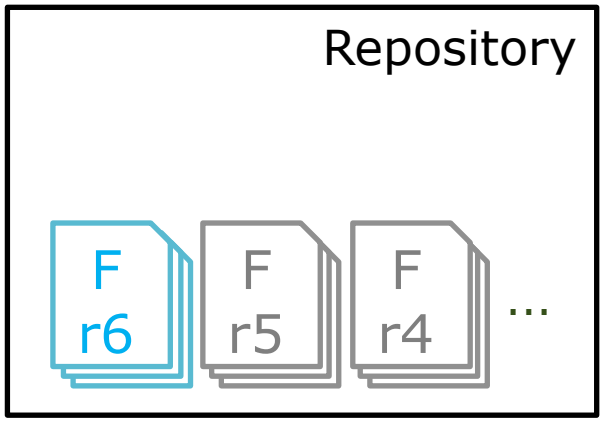
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- **Revert**: drop the modification on the working copy and reset the files to HEAD

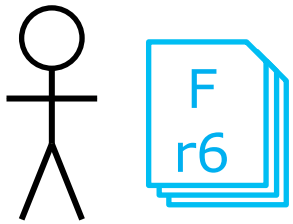


`svn revert <file>`

Main operations

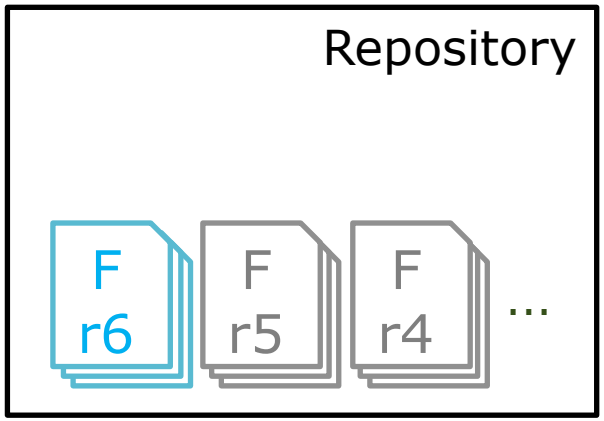
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- **Revert**: drop the modification on the working copy and reset the files to HEAD

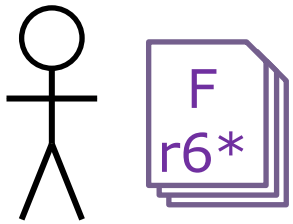




Main operations

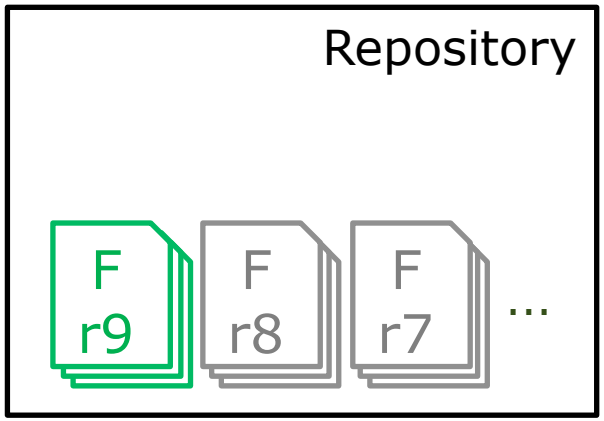
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- Revert: drop the modification on the working copy and reset the files to HEAD

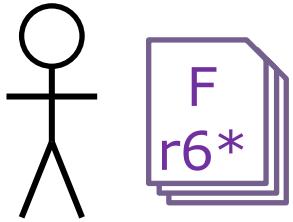




Main operations

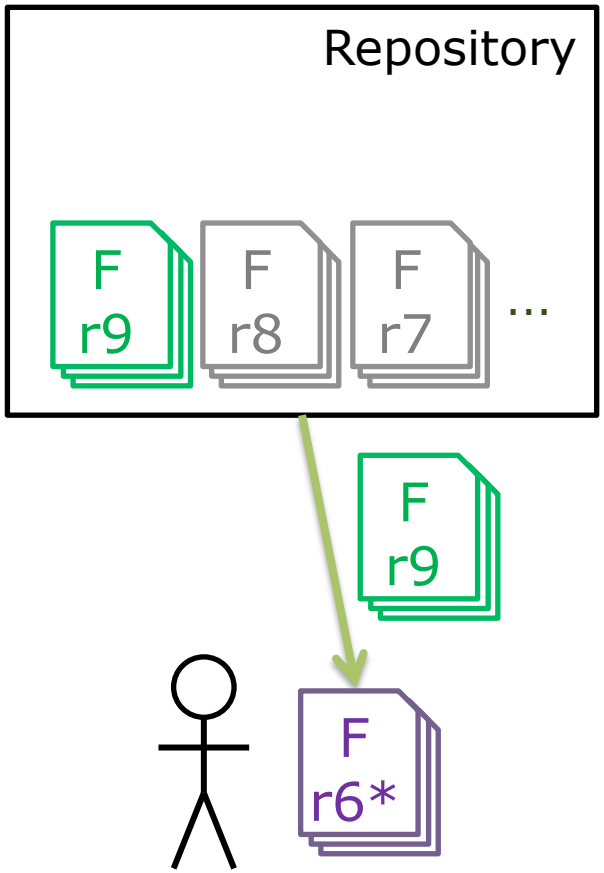
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- Revert: drop the modification on the working copy and reset the files to HEAD





Main operations

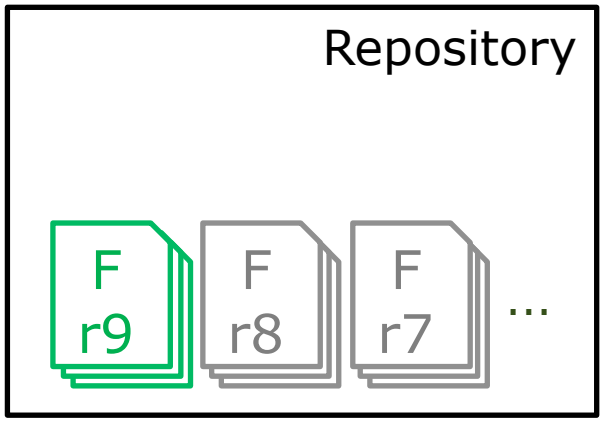
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- Revert: drop the modification on the working copy and reset the files to HEAD
- **Update**: merge HEAD and the working copy

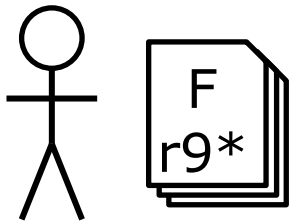


`svn update`

Main operations

- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- Revert: drop the modification on the working copy and reset the files to HEAD
- **Update**: merge HEAD and the working copy





Main operations

- The main commands we see in the previous slides are
  - `svn checkout <URL>`
  - `svn commit`
  - `svn revert <file>`
  - `svn update`
- Two additional important commands are:
  - `svn add <file> [<file>...]`
  - `svn delete <file> [<file>...]`
- `add` and `delete` respectively adds and removes files in/from the working copy
  - Those operations should then be confirmed (with a commit) or cancelled (through a revert)

Version Control System rules (3)

3. You are the only one interested in your configurations and in your output folder

- You do not have to add everything to the working space. Do not commit:
  - The settings of your IDE (e.g., .classpath, .project, .settings)
  - The configuration of the code you are writing (but you can share a configuration sample)
  - The output folder, the compiled files, etc. (e.g., bin, target)
- You may commit:
  - The source code
  - The resources that are of interest for the whole team

Conflicts

- A conflict occours when the system is unable to automatically merge a working copy with the HEAD revision
- Usually this issue can be found when developing in team
- Example:
  - Two users check-out the same release
  - They both modify the same files in their working copy
  - The first user commits his working copy
  - The second one updates and the system is unable to merge
- Before committing, you must update and resolve the conflict locally!

Version Control System rules (4)

4. Commit early, commit often and don't spare the horses

- Each commit is a "check point"
- Less conflicts and less difficulties in manage them
- The more you wait, the worse it is (for you and for your team)
- Don't be shy and/or scared

5. Commit often, but think before acting

- Commits influences the rest of the team
- Commit only when it compiles!
- Always inspect your changes before committing

Branches

- A branch is a copy of the project
  - The original is stored in the Trunk
- It is maintained separately

- There is only one trunk, while there are zero or more branches
  - Example: team works in separated branches to include new features
- SVN doesn't have a dedicated command to create branches:

```
svn copy <from> <to>
```

Merges

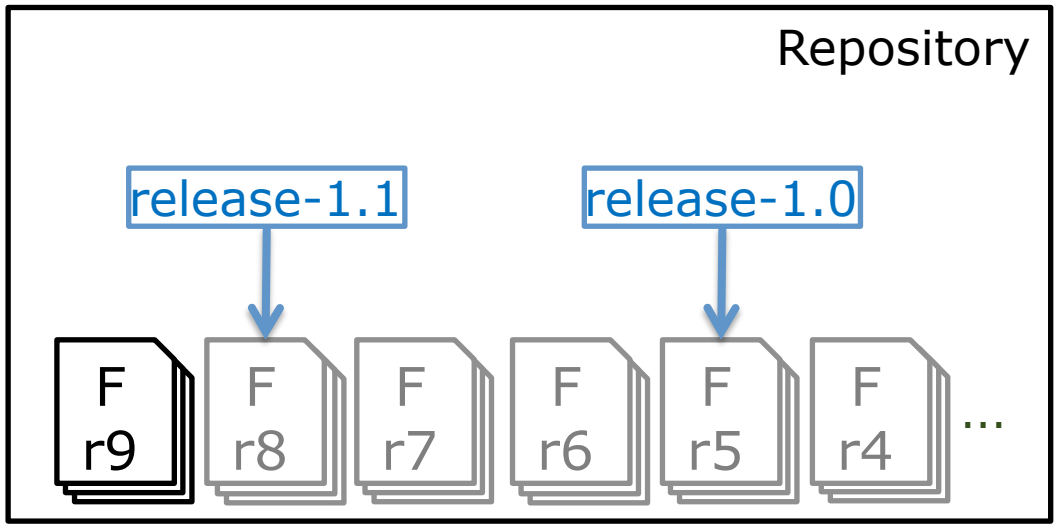
- Modifications done in the branches should then be applied to the trunk/other branches
- In general, `merge` in SVN:
  - Compares two different trees
  - Extracts the differences between the two trees
  - Differences are applied to the working copy
- The merge command is

```
svn merge -r <from>:<to> <url>
```

- As other operations, merge is done locally
  - It should be committed (or reverted)

Tags

- Tags identify relevant revisions
- Each tag is a label associated to a revision
- Tags can be used to identify
  - Milestones
  - Software releases



Version Control System rules (summary)

1. If it is not under version control, it doesn't exist
2. Remember the axe-murderer when writing commit messages
3. You are the only one interested in your configuration and in your output folder
4. Commit early, commit often and don't spare the horses
5. Commit often, but think before acting

Distributed Version Control Systems

- Distributed version control systems are an alternative to the (centralized) version control systems
- They adopt a peer to peer approach instead of a server-client one
  - Several distributed repositories
- Examples of distributed version control systems
  - Git
  - Mercurial
  - Baazar

Where are the Version Control Systems hosts? (1)

- There are several Web sites that allow you to have a SVN server
  - SourceForge http://sourceforge.net/
  - Google Code http://code.google.com/
  - Assembla https://www.assembla.com/
  - (MS) CodePlex http://www.codeplex.com/
- Download the SVN server and install it somewhere
  - It works as a service

Where are the Version Control Systems hosts? (2)

- If you want to try something different…
  - GitHub (Git) https://github.com/
  - BitBucket (Git/Mercurial) https://bitbucket.org/
  - Launchpad (Bazaar) https://launchpad.net/
- Usually (at least partially) free and for open source projects
  - Special plans for educational purposes

Useful links

- Version Control with Subversion
  http://svnbook.red-bean.com/
- A Visual Guide to Version Control
  http://betterexplained.com/articles/a-visual-guide-to-version-control/
- KDE Policies/Commit Policy
  http://techbase.kde.org/Policies/SVN_Commit_Policy
- The 10 Commandments of Good Source Control Management
  http://java.dzone.com/articles/10-commandments-good-source
- Pro Git http://git-scm.com/book