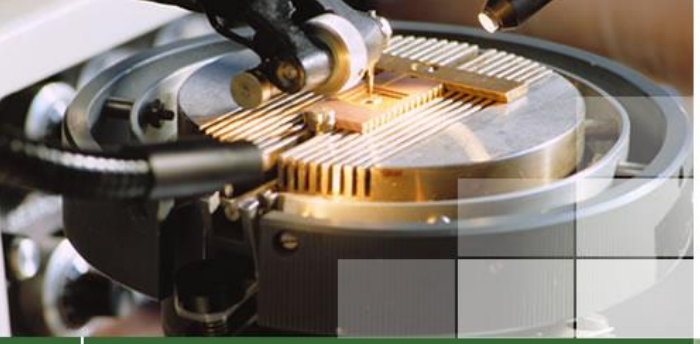


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

CEFRIEL
FORGING INNOVATION
POLITECNICO DI MILANO

Planning and Managing Software Projects 2011-12

Revision control systems

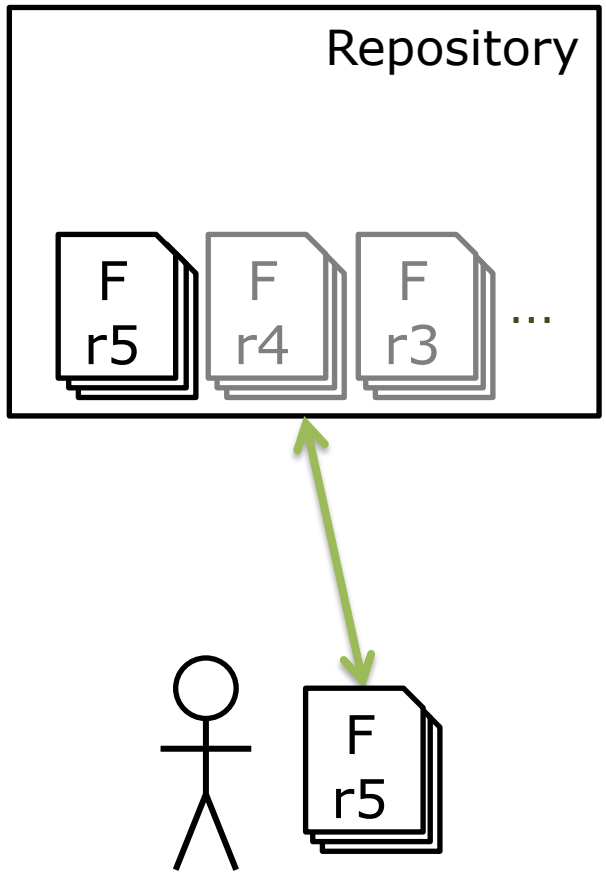
Emanuele Della Valle, Lecturer: Daniele Dell'Aglio
http://emanueledellavalle.org

Outline

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

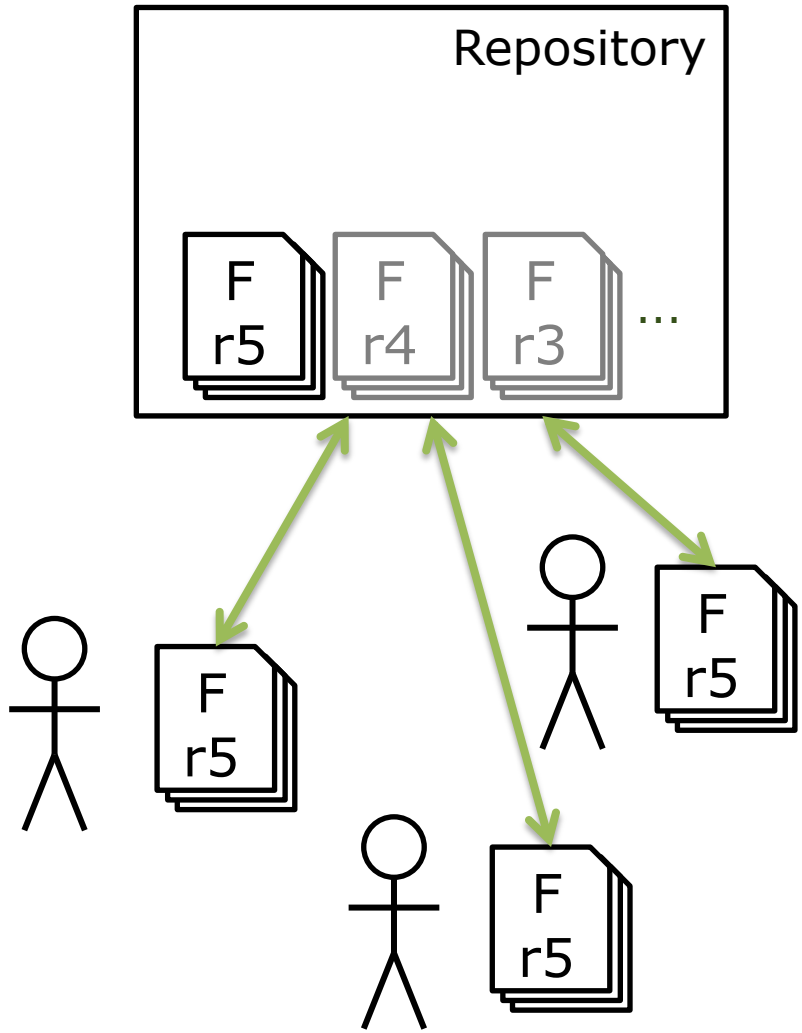
Motivations

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



Motivations

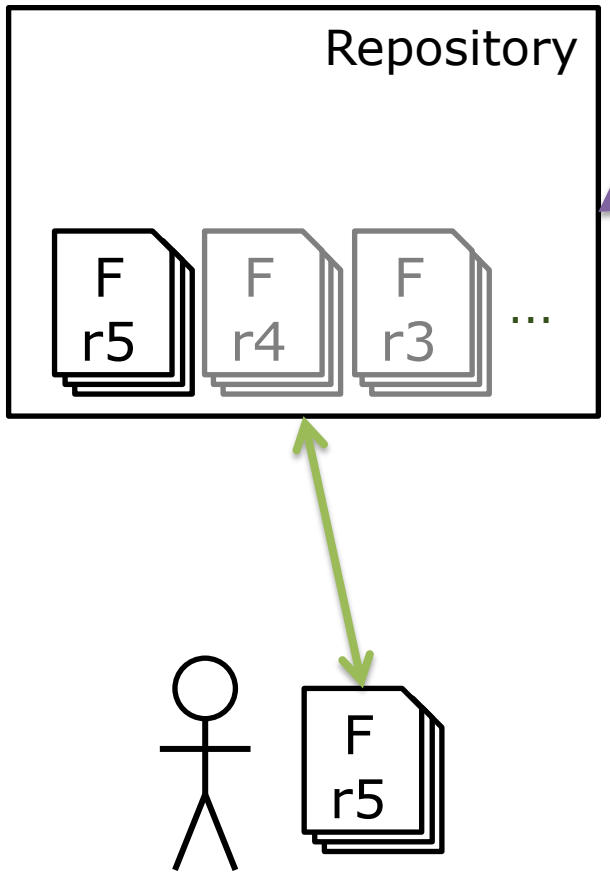
- When working alone
  - Backup copy
  - We don’t want to lose the history of our code/documents/etc.
- When working in team
  - Share modifications easily
  - Support for cooperative code development



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)
  - ...

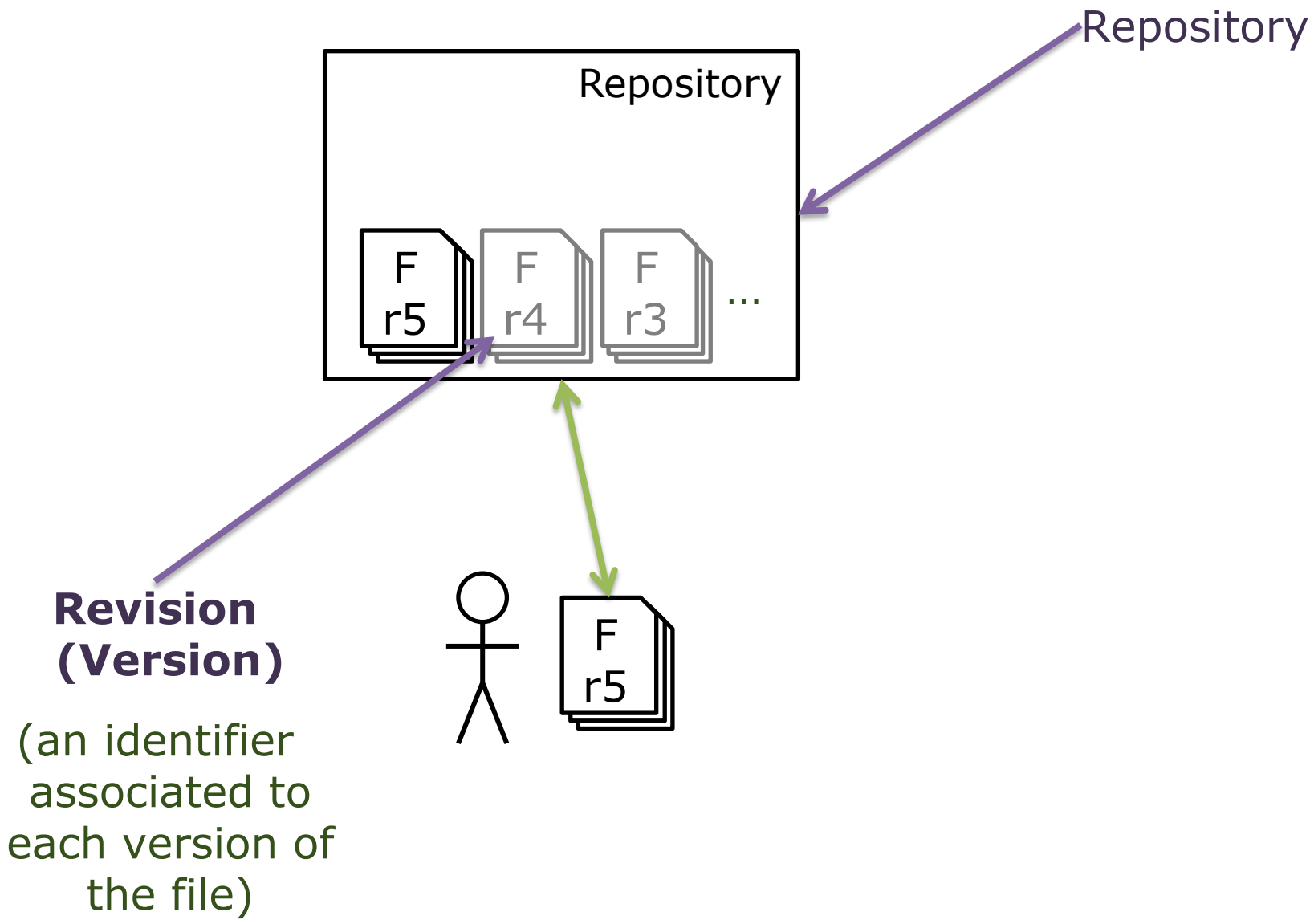
Some definitions



Repository

(the location where files are stored)

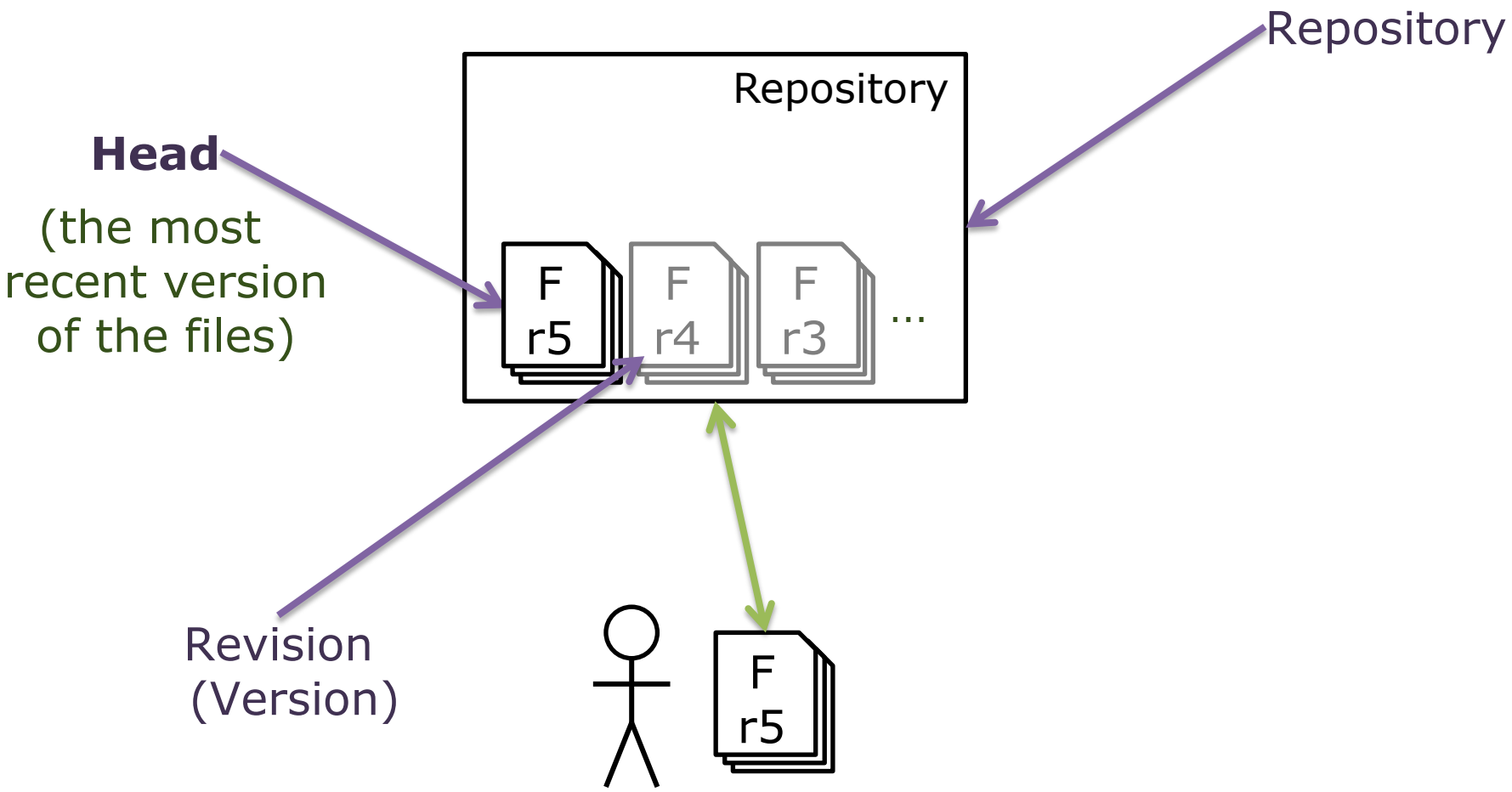
Some definitions



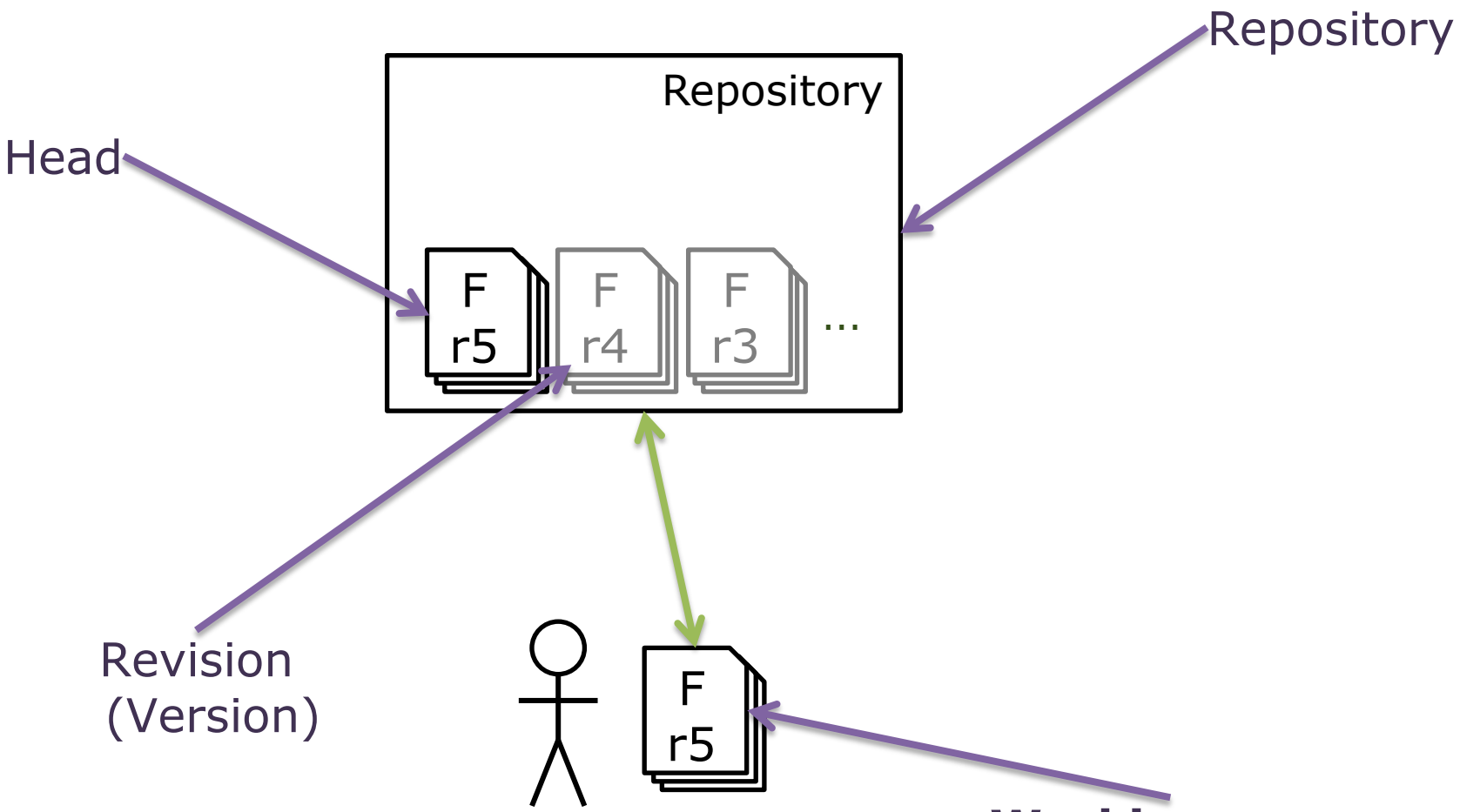
Revision (Version)

(an identifier associated to each version of the file)

Some definitions

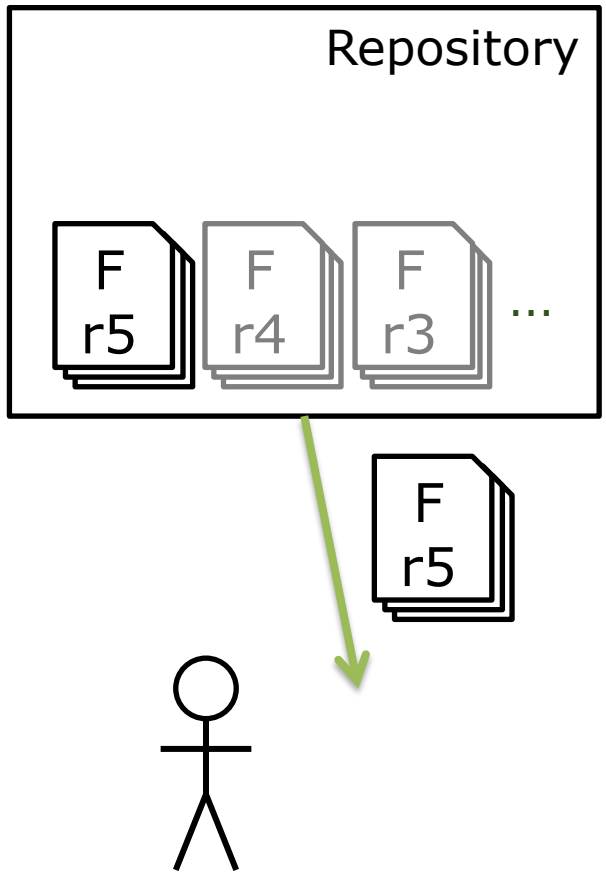


Some definitions



Main operations

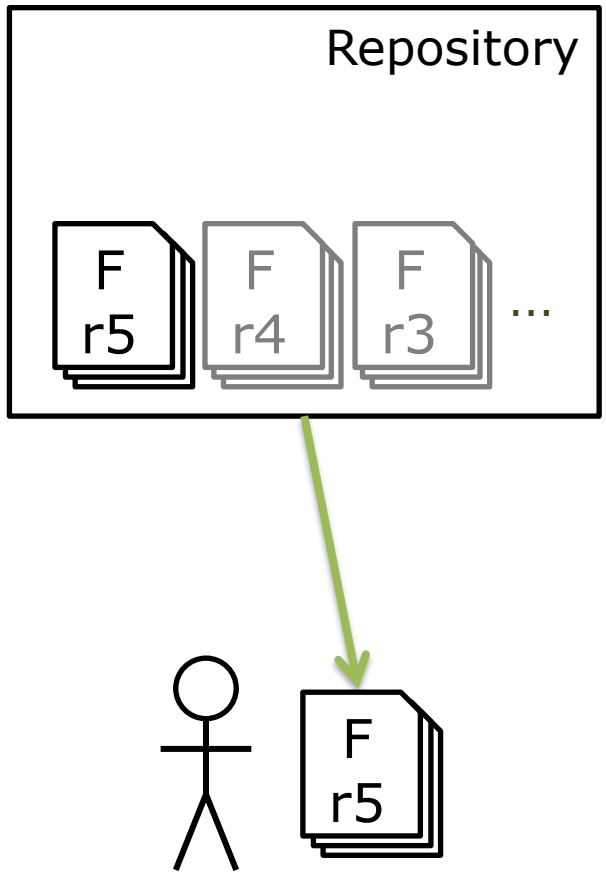
- **Check-out**: initial creation of the working copy from the repository




```
svn checkout <URL>
```

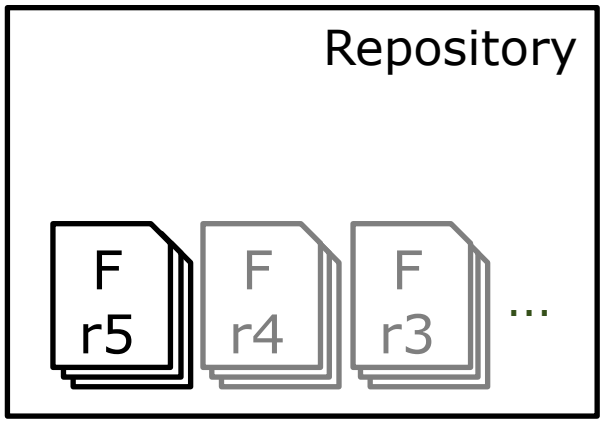
Main operations

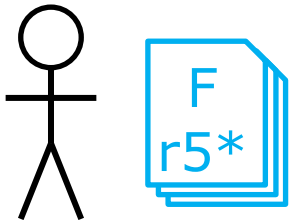
- **Check-out**: initial creation of the working copy from the repository



Main operations

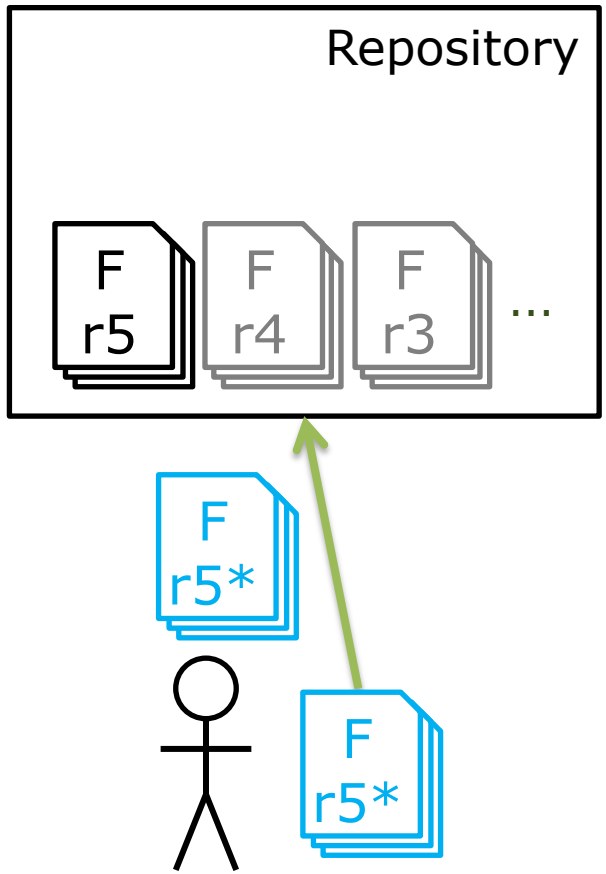
- **Check-out**: initial creation of the working copy from the repository





Main operations

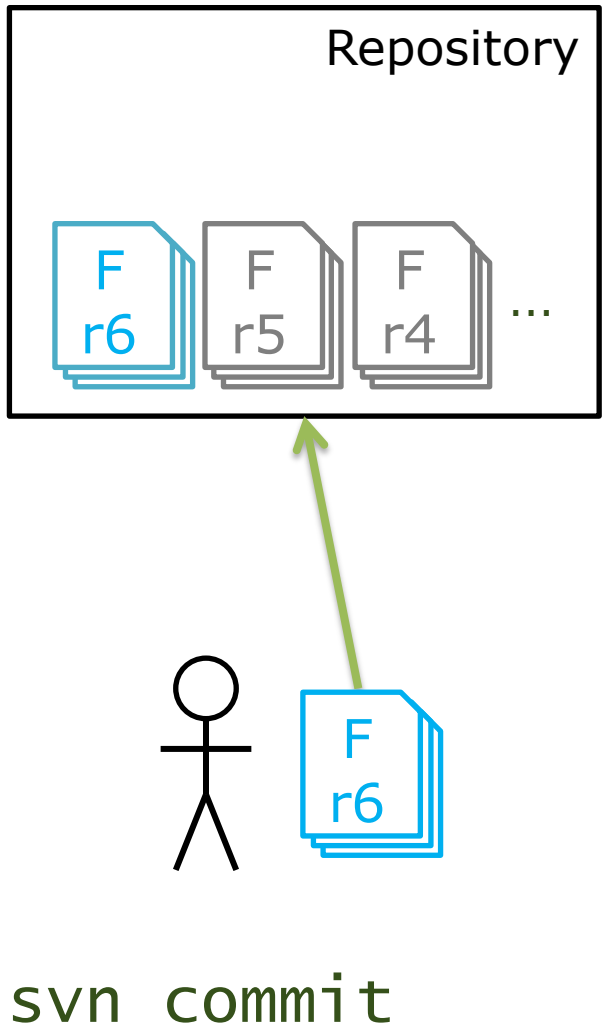
- Check-out: initial creation of the working copy from the repository
- **Check-in** (Commit): update of the HEAD revision with the working copy



`svn commit`

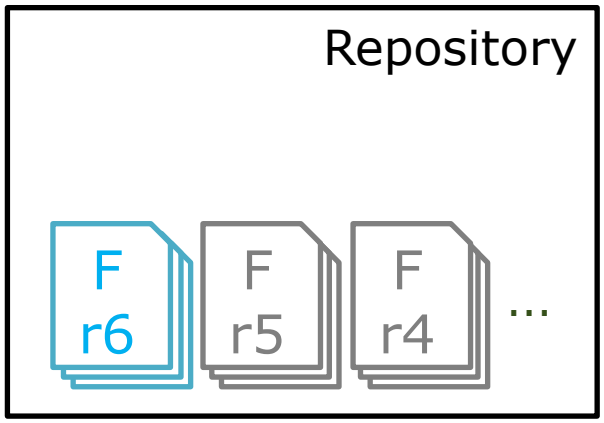
Main operations

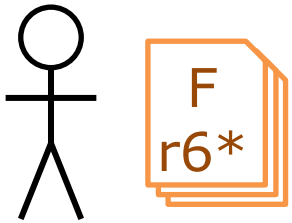
- Check-out: initial creation of the working copy from the repository
- **Check-in** (Commit): update of the HEAD revision with the working copy



Main operations

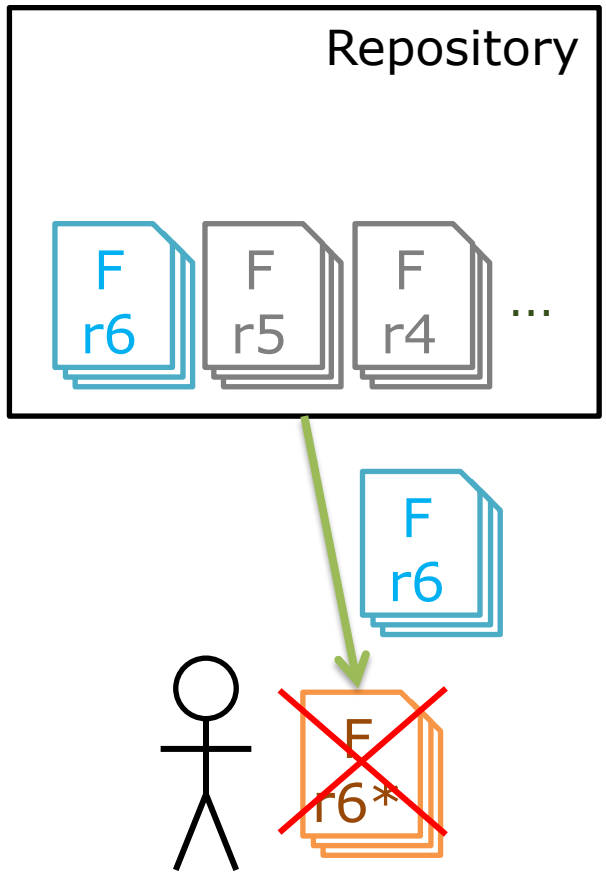
- Check-out: initial creation of the working copy from the repository
- **Check-in** (Commit): update of the HEAD revision with the working copy





Main operations

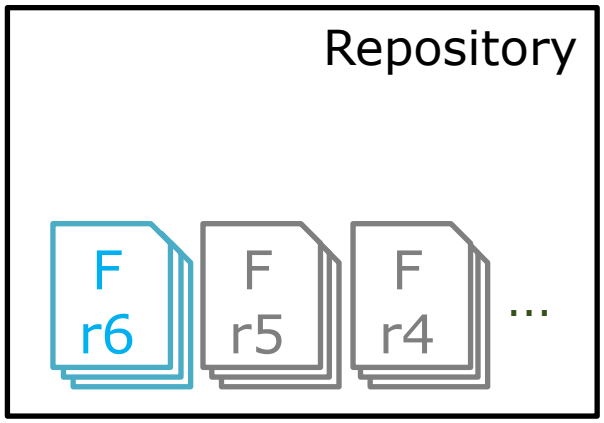
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- **Revert**: drop the modification on the working copy and reset the files to HEAD

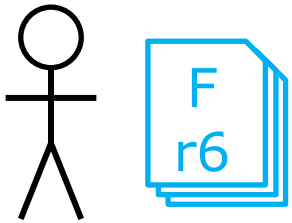


`svn revert <file>`

Main operations

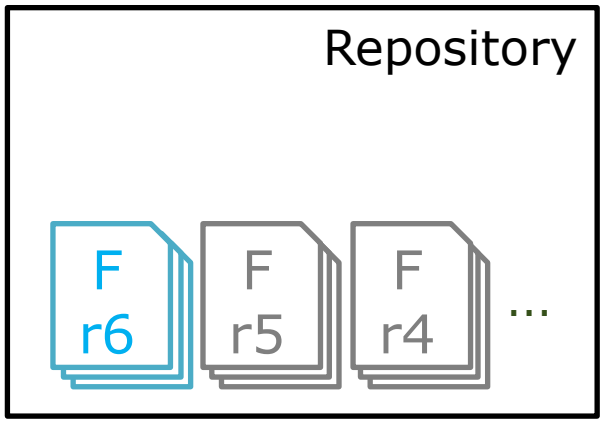
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- **Revert**: drop the modification on the working copy and reset the files to HEAD

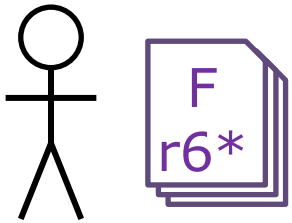




Main operations

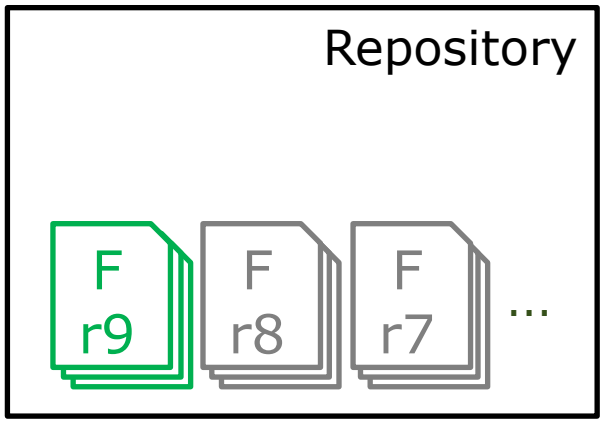
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- **Revert**: drop the modification on the working copy and reset the files to HEAD

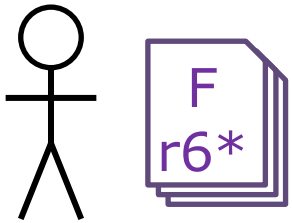




Main operations

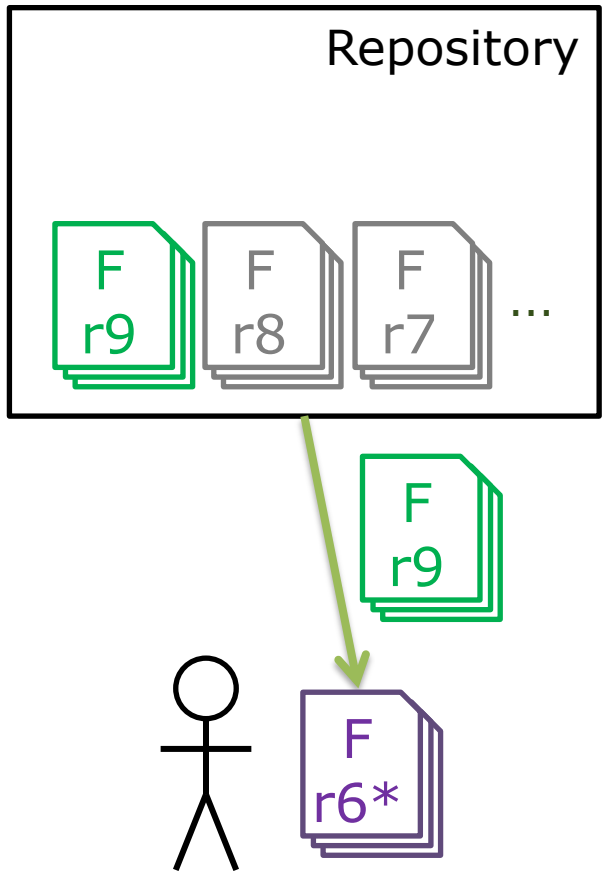
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- Revert: drop the modification on the working copy and reset the files to HEAD





Main operations

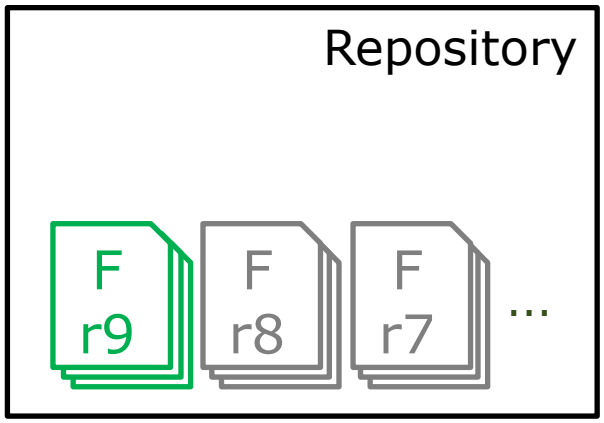
- Check-out: initial creation of the working copy from the repository
- Check-in (Commit): update of the HEAD revision with the working copy
- Revert: drop the modification on the working copy and reset the files to HEAD
- **Update**: merge HEAD and the working copy

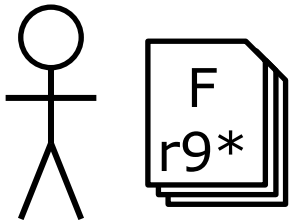


`svn update`

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)

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 tries to commit and the system is unable to merge
- Best practice: before committing, do an update and resolve the conflict locally!

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

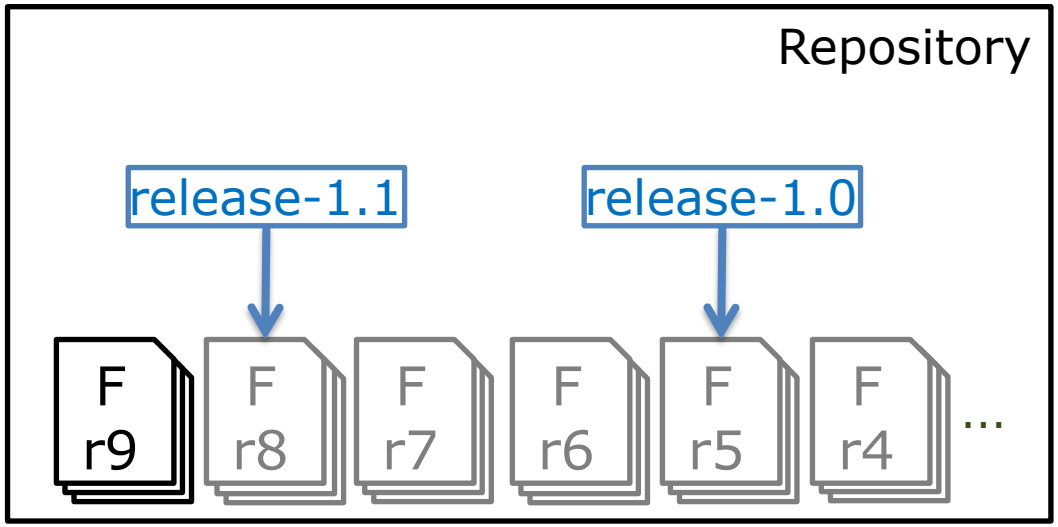
- Modifications done in the branches should then be applied to the trunk/other branches
- In general, `merge` in SVN:
  - Compares two different trees
  - Extract the differences between the two trees
  - Differences are applied to the working copy
- The merge command is

```
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



Distributed version control systems

- Distributed version control systems are an alternative to che (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

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/
- Pro Git http://git-scm.com/book