SCM
Source Code Management

Fabien Spindler

http://www.irisa.fr/lagadic

June 26, 2008

Overview

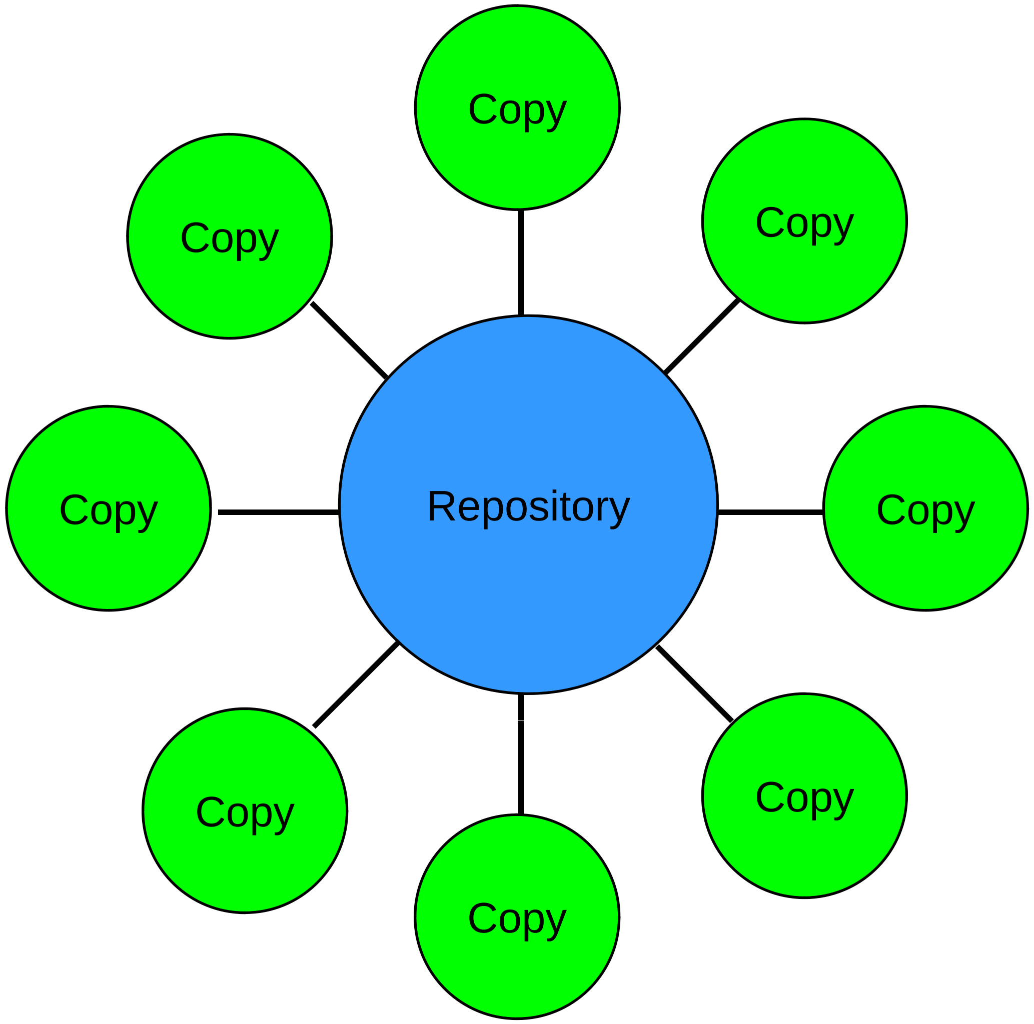
1. Application and interest

- Project source code
- Documentation
  - Manuals
  - Repports
  - Web pages
- Test
- Data

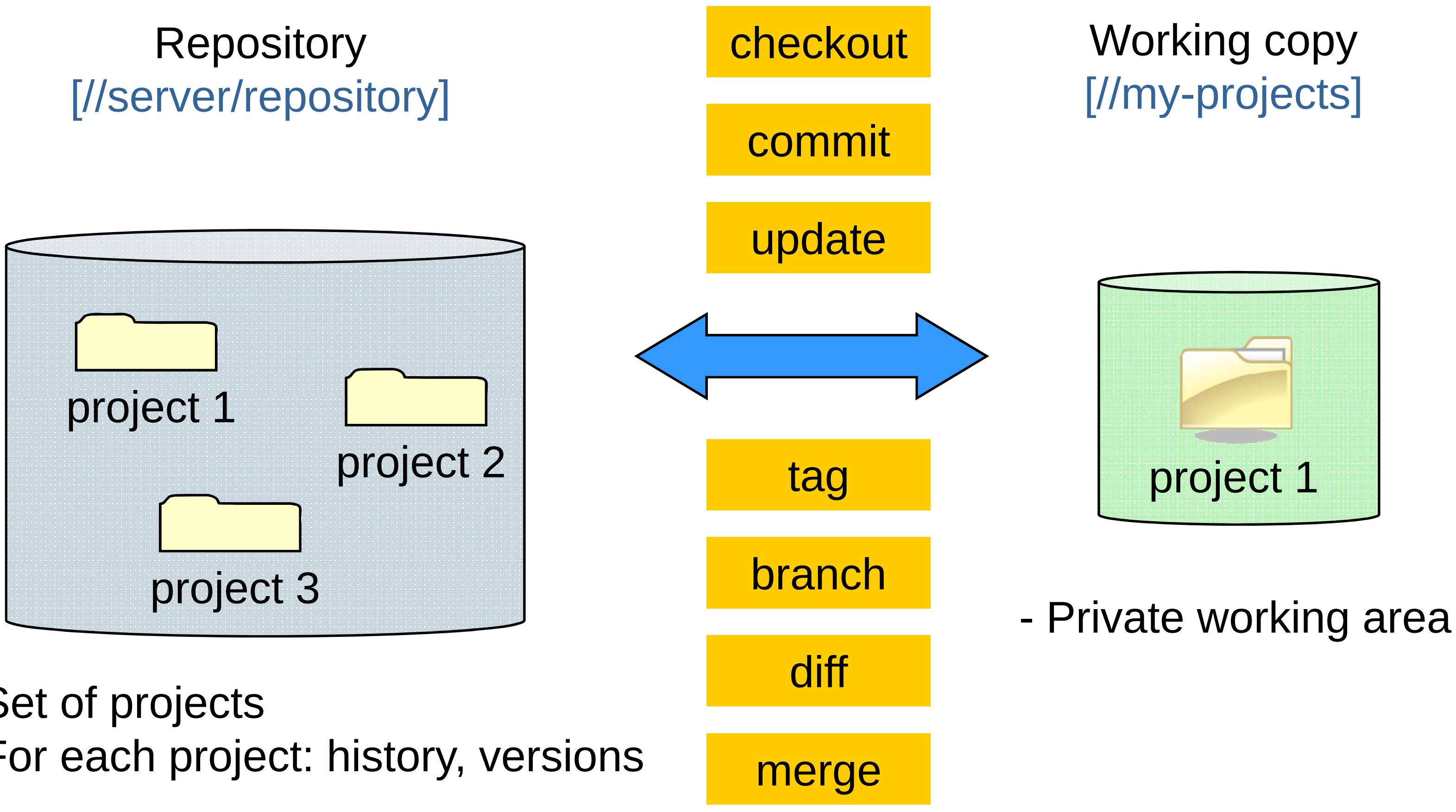
- Software development
  - Isolated / team
  - Multiple sites (laptop - /udd)
- Evolution / history management
  - Bug corrections
  - New functionalities
  - New variants / versions
  - Preserve previous versions
- Software
  - « Temporary » (phd)
  - Long-term (platform)
  - Transfert (contract)

- Motivate improvements and new versions creation
- Control the concurrent access to resources

2. Centralized source code control



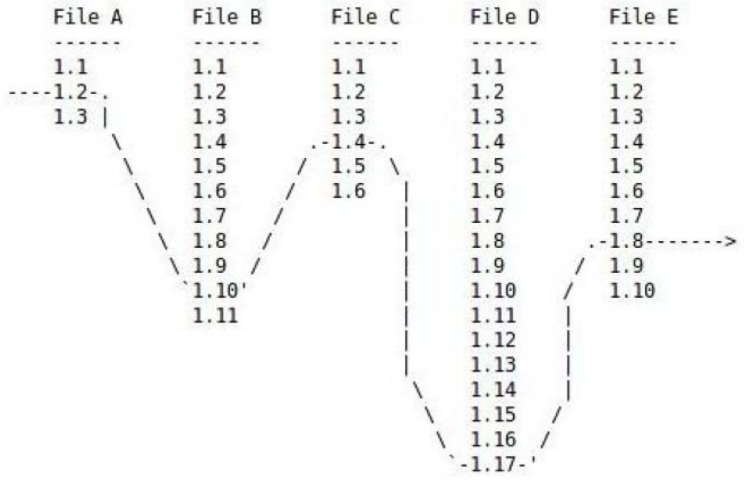
Bases



- Private working area

- Set of projects
- For each project: history, versions

CVS



```
   File A        File B        File C        File D        File E
   ------        ------        ------        ------        ------
   1.1           1.1           1.1           1.1           1.1
----1.2-.        1.2           1.2           1.2           1.2
   1.3  |        1.3           1.3           1.3           1.3
         \       1.4         .-1.4-.         1.4           1.4
          \      1.5        /  1.5  \        1.5           1.5
           \     1.6       /   1.6   |       1.6           1.6
            \    1.7      /          |       1.7           1.7
             \   1.8     /           |       1.8         .-1.8------->
              \  1.9    /            |       1.9        /  1.9
               `-1.10-'              |       1.10      /   1.10
                 1.11                |       1.11      |
                                     |       1.12      |
                                     |       1.13      |
                                      \      1.14      |
                                       \     1.15     /
                                        \    1.16    /
                                         `-1.17-'
```

- Has been very widely used
- Source code control unit is the file
- No atomic commit
- Difficult to use branches
- Sometimes difficulties for merging
- No mechanism for renaming (remove + add = history discontinuity)
- No version control for directories and data
- Not usable without network (connection to the server even for cvs diff)

Deprecated !

Subversion (SVN)

- Some commands usable without network (diff)
- Versioning on files, directories and data (properties)
- Capabilities for renaming or moving elements
- Atomic commit
  - Only if operation succeed
  - A revision number by commit (not for each file, no need to tag)
- Branches and tags: Operation deal like a copy (svn copy)
  - Each copy is a tag
  - A commit on a copy leads to a branch
- Good practices: for each project define 3 directories

trunk	(current version)
tags	(releases)
branches	(variant)

3. Getting started with Subversion

- Creating a repository

```
svnadmin create $HOME/svn
```

$HOME/svn is the location of the repository

Commands will refer to the repository as
file://$HOME/svn (Unix)
file:///Z:/svn (Windows)

- Preparing the repository for your project and files

```
svn mkdir –m “Creating project dir” file://$HOME/svn/project1
```

```
svn mkdir –m “Creating trunk dir” file://$HOME/svn/project1/trunk
```

- Importing an existing directory of files to the repository

```
svn import $HOME/project1 file://$HOME/svn/project1/trunk
```

```
Adding $HOME/project1/myImage.cpp
Adding $HOME/project1/myOldClass.cpp
```

3. Getting started with Subversion

- Listing files in the repository

```
svn list file://$HOME/svn/project1/trunk
```

$HOME/svn is the location of the repository

```
myImage.cpp
myOldClass.cpp
```

List of files in the trunk

- Creating a working copy of a project from the repository

```
mv project1 project1.bak
```

```
svn checkout file://$HOME/svn/project1/trunk project1
```

```
A          project1/myImage.cpp
A          project1/myOldClass.cpp
```

List of files that were Added to the working copy

```
cd project1
```

Next svn command are launched in the working directory

Getting started with Subversion

- Adding revised versions of files to the repository

```
svn commit  -m “my first improvement” myImage.cpp
```

-m option adds a log

- Getting logs

```
svn log myImage.cpp
```

```
r1 my first improvement
```

svn log shows the log for all the project files

- Finding out the status of your working directory files

```
svn status
```

```
?           myNewClass.cpp
M           myImage.cpp
```

? : File not under version control
M: Modified version of the file

```
svn add myNewClass.cpp
```

```
svn commit
```

Getting started with Subversion

- Extracting updated versions of files from the repository

```
svn update
```

- Renaming or deleting files

```
svn rename myNewClass.cpp myClass.cpp
```

```
A          myClass.cpp
D          myNewClass.cpp
```

Added file
Deleted file

```
svn delete myOldClass.cpp
```

```
D          myOldClass.cpp
```

```
svn commit
```

```
Adding   myClass.cpp
Deleting myNewClass.cpp
Deleting myOldClass.cpp
```

Getting started with Subversion

- Resolving conflicting version of a file

```
svn commit
svn: Commit failed
svn update
```

There is a problem due to a conflict

Try to solve the conflict

1. Merge the two conflicting versions into a combined version with success

```
G          myClass.cpp
svn commit
```

MerGe was done with success

2. Merge failed

```
C          myClass.cpp
myClass.cpp.r7
myClass.cpp.r8
myClass.cpp.mine
```

There is a Conflict.
svn produces 3 extra files

```
myClass::myClass() {
<<<<<<< .mine
  dummy1();
=======
  dummy2();
>>>>>>> .r8
```

Release 7, checked out and edited twice
Release 8, checked in from an other working copy
Version in the dir that conflicts with the repository

```
svn resolved myClass.cpp
svn commit
```

Suppress also the 3 extra files

Getting started with Subversion

- Reverting your working copy back to an earlier version from the repository

`svn revert myClass.cpp`	Replace with the most recent commited version
`svn update –r 6 myClass.cpp`	Replace with revision 6 from the repository

- Looking at old versions of files without reverting them

```
svn cat -r 3 myClass.cpp
```

Getting started with Subversion

- Getting the differences between the working directory and the repository

```
svn diff
```

- Creating a patch

```
svn diff –r10:21 *.cpp *.h > /tmp/r10-to-r21.patch
```

- Updating a ChangeLog file automatically

```
svn log #10:HEAD >> ChangeLog
```

```
svn commit ChangeLog
```

Get all the changes since revision #10
HEAD refers to the last revision number

- Tags and branches

```
svn copy file://$HOME/svn/project1/trunk file://$HOME/svn/project1/tags/project1-2.0
```

The repository has now 2 main subdirs
project1/trunk and project1/tags/project1-2.0
They can now evolve separately. If no commit is done on project1-2.0 it is called a tag. Otherwise it is a branch.

Getting started with Subversion

- Creating a distribution

```
svn export file://$HOME/svnrepos/project1/tags/project1-2.0
```

```
tar cvzf project1-2.0.tar.gz project1-2.0
```

- svn access via ssh (laptop - /udd)

```
svn list svn+ssh://username@nereide/udd/username/svn
```

username has an ssh acces to nereide computer
where the repository is located in /udd/username/svn

4. Conclusion

- ☐ Centralized SCM

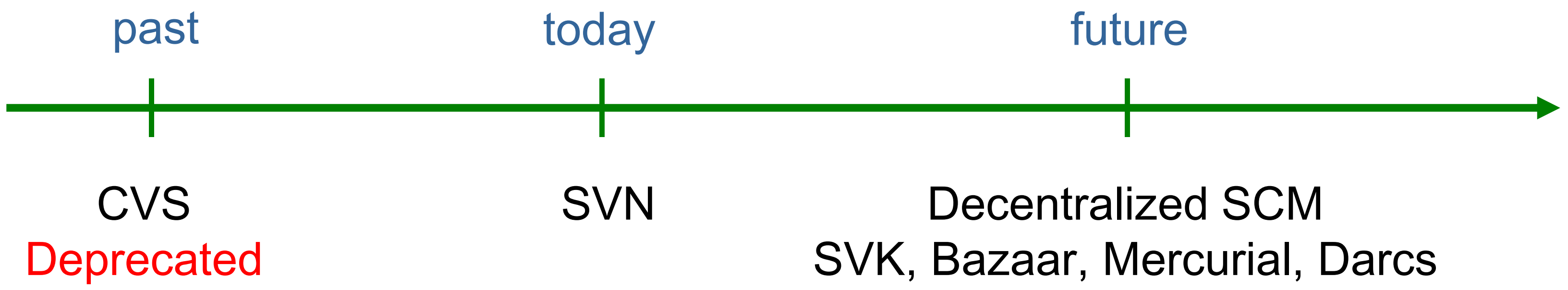
  + Very useful for software development, especially Subversion

  + Reference repository

  + Simple to use

  - Need to be connected to the server for some commands

  - Need to have specific privilege (commit)



- ☐ ViSP: migration from CVS to SVN before the next release