

# Versioning systems tutorial

## with SVN

Samuel Colin<sup>1</sup>

LORIA – Université Nancy 2

<sup>1</sup> Campus Scientifique, BP 239

F-54506 Vandœuvre-lès-Nancy cedex

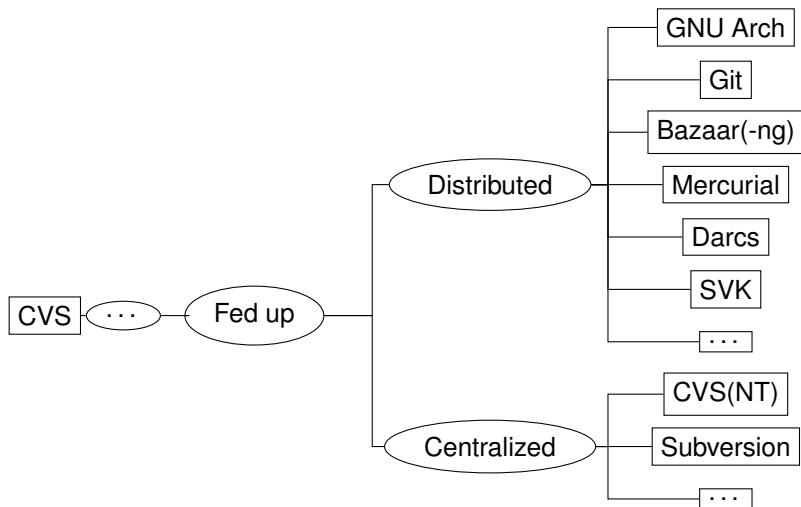
Email: {firstname.lastname}@loria.fr

LORIA/DEDALE

# Outline

- 1 Introduction to VSeS
  - A bit of history
  - Typical work with a VS
  - Closing words about VSeS
  
- 2 Using SVN
  - A typical work with SVN
  - Tutorial

# History





## What does a VS do ?

- Keeps track of modifications: who, what, when, how ?
- Gives a way for several developers to intervene on a same project at the same time
- Gives tools for solving contradicting modifications (conflicts)
- Identifies interesting versions of the project
- Eases more thorough changes of the project (branching)...
- ... and the reuniting with the official version of the project (merging)

## The typical workflow

- Checkout of the repository (*I want to contribute*)
- Loop:
  - Update (*what did others change ?*)
  - Modify
  - Update + conflicts resolution
  - Commit (*I validate my changes*)
  - Repeat Loop

## References

- [http://en.wikipedia.org/wiki/Comparison\\_of\\_revision\\_control\\_software](http://en.wikipedia.org/wiki/Comparison_of_revision_control_software)

## A few last words

- A versioning system is *not* a backup system
- A versioning system does not replace communication
- A best practice is to leave the central repository in a compilable state

# Outline

- 1 Introduction to VSeS
  - A bit of history
  - Typical work with a VS
  - Closing words about VSeS
- 2 Using SVN
  - A typical work with SVN
  - Tutorial

# The typical workflow:

the actual commands

- `svn checkout protocol://directory local-directory`
- `cd local-directory`
  - `svn update`
  - **Modify. Can be:**
    - `svn add some-file`
    - `svn remove file`
    - `svn move file1 file2`
    - `svn copy some-file`
    - **Modify files**
  - `svn update (there might be conflicts, see later)`
  - `svn commit`
  - **Repeat**

`svn help` for details about the commands

# The typical workflow:

## cases of conflicts

- On an update: It will look like this

```
C    test.txt
Actualisé à la révision 2.
```

- On a commit: Subversion complains

```
Envoi      test.txt
Transmission des données .svn: Échec de la propagation (commit), détails :
svn: Le chemin 'test.txt' est obsolète dans la transaction '3-1'
```

**Solution:** do an update

We shall next see how to resolve a conflict.

## Conflicts:

what has happened ?

svn status **leaves us with:**

```
?      test.txt.r2
?      test.txt.mine
?      test.txt.r1
C      test.txt
```

## Conflicts:

what has happened (cont'd)

What are these ?

**test.txt.r2** The version of the central repository

**test.txt.mine** My version has I attempted to commit it

**test.txt.r1** The last version up-to-date with the central repository

**test.txt** My file, with conflicts pinpointed inside like this:

```
<<<<<<< .mine
Youhou !
=====
Awesome !
>>>>>>> .r2
```

# Conflicts:

## resolving

- Replace the part between <<<<<< and >>>>>> with something relevant
- Repeat the previous step for all conflicts inside a file (there might be several !)
- Repeat the previous step for all conflicting files
- `svn resolved conflicting-file(s)`

Finally you can commit your changes ! (and there might be some more conflicts)

# Subversion:

## other features

- Automatic variables (special variables replaced by the date of last modification, version, etc)
- Tagging (tag a version of specific interest)
- Branching (i.e. forking the development in order not to disturb other developers)
- Merging (contributing back the changes made in a branch)

## A few references

- [Collins-Sussman et al., 2007]
- `svn help`
- <http://wiki.loria.fr/wiki/Subversion%40dedale> (**see the part with automatic variables**)

## Creating a repository, both central and local

- Create the central repository (the tutorial monitor has already done this)
  - `umask 007` (group-based authorization, Unix filesystem)
  - `svnadmin create SVN-test` (path: `/users/dedale/colinssa/pub-dedale/`)
- Checkout out the repository and get ready
  - `cd some-directory`
  - `svn co svn+ssh://login@greny/users/dedale/colinssa/pub-dedale/SVN-test svn-tutorial`
  - `cd svn-tutorial`

## How the tutorial will be held

- We will build a LaTeX document presenting each member of the tutorial session
- Because we use various languages and because the session monitor is perverse, we will use Unicode as the input encoding (Emacs users need not worry)
- To each person will correspond a section he/she is responsible of
- The content of each section will be saved in a file
- Each file will be an input for the  $\LaTeX$ main file



```
% -*- coding: utf-8 -*-  
\documentclass{article}  
  
\usepackage[utf8x]{inputenc}  
\usepackage[T1]{fontenc}  
\usepackage{pslatex}  
  
\title{SVN tutorial session}  
\author{DEDALE}  
  
\begin{document}  
\maketitle  
  
\begin{abstract}  
\end{abstract}  
  
\end{document}
```

Emacs:  $\LaTeX$  → Multifile/parsing → Set master file

## Follow this outline

- Write your own section in your own file (find a consistent naming, put `␣ -*- coding: utf-8 -*-` at the beginning)
- Add it, commit your change
- Reference it in the main file
- Commit your change
- Two persons will write the abstract independently and commit
- One person will add a french(?) translation for each section
- One person will make an introduction presenting the various sections
- One person will spell-check everything
- Get to work !

## Don't forget

- svn update
- **Modify** (add, move, remove, copy or file content modifications)
- svn update
- **If there are conflicts:**
  - svn status to see the conflicting files (C in the left column)
  - Edit the offending file(s)
  - svn resolved offending-file for each file
- svn commit
- **Note:** when committing, *use a good commit message*

## What happened so far ?

Two things might have happened:

- The repository is now a mess and people are still discussing about how to correct things  
⇒ You have not worked enough. Go on !
- The repository is super-duper-clean, and a hierarchy has arisen  
⇒ Congratulations, you just understood (if subconsciously) how to get things done with a versioning system

## Of interest to those who dislike command-line

There are graphical interfaces to Subversion

- TortoiseSVN
- RapidSVN
- SCPlugin
- SmartSVN
- ...

- See `http:`

`//en.wikipedia.org/wiki/Subversion_%28software%29`



## Next time

- Tagging
- Branching
- Merging
- Special properties

# References



Collins-Sussman, B., Fitzpatrick, B. W., and Pilato, C. M. (2007).

**Version control with subversion.**

website.

<http://svnbook.red-bean.com/>.