# Tools for collaboration

## Decoda meeting

4 November 2015

# Outline

Redmine

Repositories (version control)

GIT for paper writing

# Overview

Redmine

Repositories (version control)

GIT for paper writing

## Some facts

- Open-source project management platform
- Web-based interface
- Developed since 2006
- Many developers from France (main developer: Jean-Philippe Lang)
- Now has a GIPSA installation

# GIPSA-lab Redmine installation

Address: `https://redmine.gipsa-lab.grenoble-inp.fr`

https is important!

redmine.gipsa-lab.grenoble-inp.fr

Espace webcam du groupe Grenoble INP

Webcam de Grenoble INP - Campus

Grenoble INP : vue du campus de Grenoble et de Belledonne 2015-11-03 22:40:25

Vue sur le campus de Grenoble et Belledonne

# Main features

- Projects, which can contain subprojects
- Projects are public or private
- Each project/subproject contains
  - Wiki pages
  - News
  - Issue tracker
  - Activity report / notifications
  - Repositories

## Main features

- Projects, which can contain subprojects
- Projects are public or private
- Each project/subproject contains
  - Wiki pages
  - News
  - Issue tracker
  - Activity report / notifications
  - Repositories

Go to `https://redmine.gipsa-lab.grenoble-inp.fr`

# Projects and subproject

- Main project (DECODA, public) created by SI (service informatique)
- Subprojects can be created by the users with “Manager rights”
- Subprojects (private) can be accessed by different people

# Documents (Fichiers)

Just a tool for sharing files.

# Wiki

- Page(s) with special formatting
- Uses Textile and WikiNG (lists, tables, etc.)
- Supports LaTeXsyntax
- Images can be added
- Supports multiple pages
- Contains history of changes

# News

Just news

## Issues (Demandes)

- Initially designed for reporting bugs (for example, `https://redmine.gipsa-lab.grenoble-inp.fr/projects/support/issues`)
- Can open/closed/assigned to a person
- Users can comment on issues
- Can be used to manage tasks (for example, `https://redmine.gipsa-lab.grenoble-inp.fr/projects/nuclear-norm-matrix-completion/issues`)
- You can select people who will be notified

# Activity/notifications

- You receive notifications by email about issues
- You can watch Wiki pages (should be fixed soon by the SI)
- Notifications about mentions of people.
- Settings for notifications in your profile `https://redmine.gipsa-lab.grenoble-inp.fr/my/account`
- A summary of activity: for the whole project, for sub-projects... `https://redmine.gipsa-lab.grenoble-inp.fr/projects/decoda/activity`

# Repositories

Probably the main thing in Redmine!

# Overview

Redmine

Repositories (version control)

GIT for paper writing

# An example: collaborative working on a file

Suppose, you are working on a file `paper.tex`

Typical collaboration workflow:

- Exchange files by email/Dropbox
- One person can edit at a time
- Name versions of the file: `paper_01.tex`, `paper_02.tex`, `paper_final.tex`, `paper_final_v01.tex`, `paper_final_v01_corrections.tex`, ...
- Problems appear when you have $N$ people

# What if you have a set of files?

For example, set of MATLAB functions:

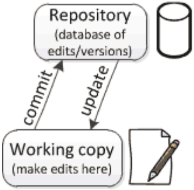
- `algebraicFit.m`
- `geometricFit.m`
- `multPolynomials.m`
- `quasiHankel.m`
- ...

Collaboration workflow:

- Exchange files by email? Versioning?
- What if we want to work simultaneously on different files

# Repository: a single user example

Repository: an organized collection of snapshots of directory

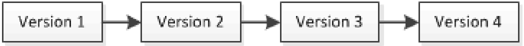


Repository
(database of
edits/versions)

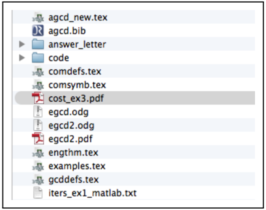
commit

update

Working copy
(make edits here)



Version 1 → Version 2 → Version 3 → Version 4



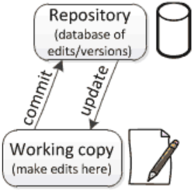
```
    agcd_new.tex
    agcd.bib
▶   answer_letter
▶   code
    comdefs.tex
    comsymb.tex
    cost_ex3.pdf
    egcd.odg
    egcd2.odg
    egcd2.pdf
    engthm.tex
    examples.tex
    gcddefs.tex
    iters_ex1_matlab.txt
```

A jargon:

- **working copy**: set of your files
- **commit/changeset**: a saved snapshot of your files
- **to commit**: create a snapshot of your data

# Repository: a single user example

Repository: an organized collection of snapshots of directory

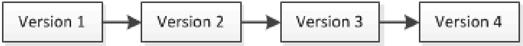


Repository (database of edits/versions)

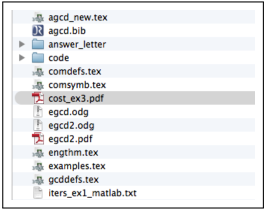
commit

update

Working copy (make edits here)



Version 1 → Version 2 → Version 3 → Version 4



agcd_new.tex
agcd.bib
answer_letter
code
comdefs.tex
comsymb.tex
cost_ex3.pdf
egcd.odg
egcd2.odg
egcd2.pdf
engthm.tex
examples.tex
gcddefs.tex
iters_ex1_matlab.txt

A jargon:

- **working copy**: set of your files
- **commit/changeset**: a saved snapshot of your files
- **to commit**: create a snapshot of your data

Show an example

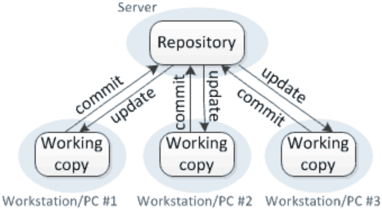
## Version control systems: multiple users

- Generation 2: centralized VCS (like Subversion, or SVN)
- Generation 3: decentralized VCS (like GIT)

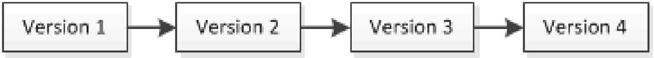
# Subversion (SVN)

- One central repository

Centralized version control



+ Only linear history is allowed: easy to start

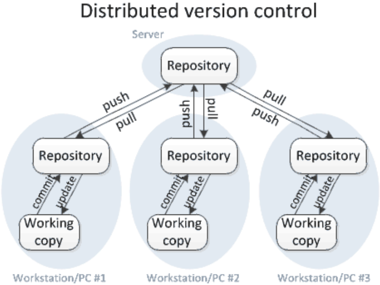


– Needs a central server

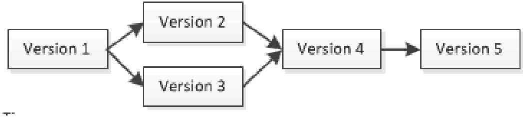
– Rigid structure

# Distributed VCS (GIT)

- Local repositories + remote repositories



+ Acyclic directed graph of changes: very flexible



+ Local commits (history can be stored locally)

– Too many features, steep learning curve

# Distributed VCS (GIT)

- Initially, command line program
- Included in the MacOS developer tools (and in Linux by default)
- Has many graphic user interfaces (GUIs)
- Many servers for hosting GIT repositories
  - GitHub — many features, but paid private repositories
  - Bitbucket, Gitlab — free private repositories
  - Now there is a GIPSA server, integrated with Redmine

# Distributed VCS (GIT)

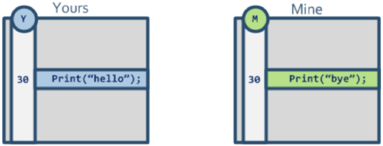
- Initially, command line program
- Included in the MacOS developer tools (and in Linux by default)
- Has many graphic user interfaces (GUIs)
- Many servers for hosting GIT repositories
  - GitHub — many features, but paid private repositories
  - Bitbucket, Gitlab — free private repositories
  - Now there is a GIPSA server, integrated with Redmine
- Show an example of changes + GIPSA server

# Resolving conflicts in one file: merging

- If changes in different files: no problem

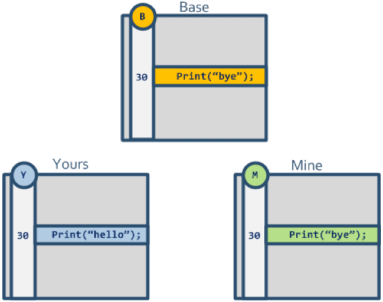
# Resolving conflicts in one file: merging

- If changes in different files: no problem
- If changes in one file?



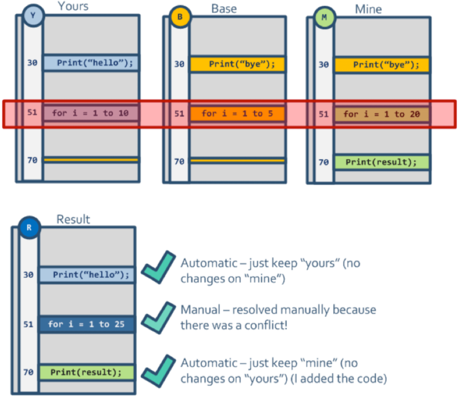
# Resolving conflicts in one file: merging

- If changes in different files: no problem
- If changes in one file? → **3-way** merge



# An example of 3-way merge

- An illustration:



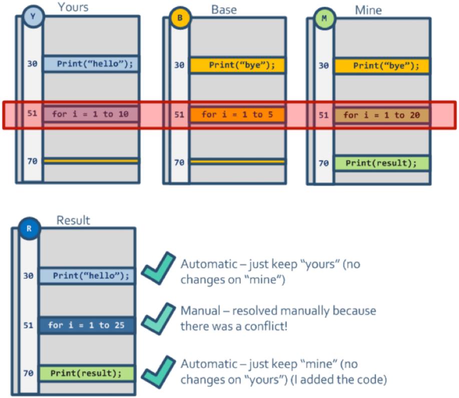
Automatic – just keep "yours" (no changes on "mine")

Manual – resolved manually because there was a conflict!

Automatic – just keep "mine" (no changes on "yours") (I added the code)

# An example of 3-way merge

- An illustration:



Yours:

```
30  Print(“hello”);
51  for i = 1 to 10
70
```

Base:

```
30  Print(“bye”);
51  for i = 1 to 5
70
```

Mine:

```
30  Print(“bye”);
51  for i = 1 to 20
70  Print(result);
```

Result:

```
30  Print(“hello”);
51  for i = 1 to 25
70  Print(result);
```

✓ Automatic – just keep “yours” (no changes on “mine”)

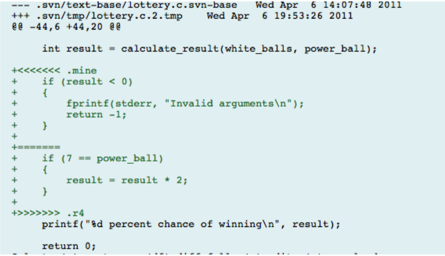
✓ Manual – resolved manually because there was a conflict!

✓ Automatic – just keep “mine” (no changes on “yours”) (I added the code)

- Show a real-life example.

# Manual conflict resolution

- Just edit the text file



```
--- .svn/text-base/lottery.c.svn-base   Wed Apr  6 14:07:48 2011
+++ .svn/tmp/lottery.c.2.tmp    Wed Apr  6 19:53:26 2011
@@ -44,6 +44,20 @@

     int result = calculate_result(white_balls, power_ball);

+<<<<<<< .mine
+    if (result < 0)
+    {
+        fprintf(stderr, "Invalid arguments\n");
+        return -1;
+    }
+
+=======
+    if (7 == power_ball)
+    {
+        result = result * 2;
+    }
+
+>>>>>>> .r4
     printf("%d percent chance of winning\n", result);

     return 0;
```

- use GUI (for example, kdiff3)

## Version control systems: usage

- A backup/synchronization tool
- Almost a must for code
- Now a standard way to distribute code: (the concept of rolling beta version)
  An example: `https://github.com/oseledets/TT-Toolbox`

# Version control systems: usage

- A backup/synchronization tool
- Almost a must for code
- Now a standard way to distribute code: (the concept of rolling beta version)
  An example: `https://github.com/oseledets/TT-Toolbox`
- A possible way of collaborative paper writing

# Overview

Redmine

Repositories (version control)

GIT for paper writing

# Collaborative paper writing

Success story: HoTT book

- 40 mathematicians collaborating on Homotopy type theory
- They use GIT (public GitHub repository)
  `https://github.com/HoTT/book`

## Personal experience

- I use it since 2013 for papers (since 2009 for software)
- Of course, private repositories

## Personal experience

- I use it since 2013 for papers (since 2009 for software)
- Of course, private repositories
- Examples:
  - IEEE TSP
  - JstatSoft
  - LVA-ICA paper
  - Nuclear norm

## Other collaborative paper writing software

Mostly paid services:

- TexPad app in Mac
- Authorea
- ShareLatex — can be installed on own server, based on GIT inside

Why should we pay for that?

# References

- A basic introduction into VCS: https://homes.cs.washington.edu/~mernst/advice/version-control.html
- An overview of SVN and GIT http://ericsink.com/vcbe/
- A detailed book on GIT http://git-scm.com/book/en/v2
- Collaborative paper writing tools: http://www.inferencelab.com/collaborative-paper-writing-in-latex/,
- With GIT: https://en.wikibooks.org/wiki/LaTeX/Collaborative_Writing_of_LaTeX_Documents https://jmanton.wordpress.com/?s=git
- Pierre Pignede is working on Redmine for CHESS — maybe we can ask for his help