COOPERATIVE SOFTWARE ENGINEERING & CONFIGURATION MANAGEMENT

Matthieu Tixier – #6

Outline

- Developing software as a team
  - The case of Open Source Software (OSS)
- Configuration Management
  - « Configuration » ?
  - Source code and version management
  - Subversion
- Discussion

Cooperative work

- From specifications to implementation
- Being several to work on a same code
  - Is it a problem?

Cooperative work

- From specification to implementation
- Being several to work on a same code
  - Is it a problem?
    - Splitting the tasks
      - --> Please wait until I'll finish my work to touch that file. May be next week...
      - --> I didn't succeed to import your code...
    - After several iterations
      - --> oups, I have seen this working before we update for v2
      - --> Why do we had to rewrite the random number generator from the standard lib two years before... any ideas?
    - Welcoming new contributors
      - --> Where am I supposed to put my code for the internship?

Cooperative work

- From specification to implementation
- Being several to work on a same code
  - Is it a problem?
    - Splitting the tasks
      - Sequence --> Please wait until I'll finish my work to touch that file. May be next week...
      - Parallel --> I didn't succeed to import your code...
    - After several iterations
      - Keeping track of the software evolution
        --> oups, I have seen this working before we update for v2
      - Understanding the rationale behind a project history
        --> Why do we had to rewrite the random number generator from the standard lib two years before... any ideas?
    - Welcoming new contributors
      - --> Where am I supposed to put my code for the internship?

➔ Preparing for software evolution, enabling maintenance work

Yes, we can

- Open source
  - Examples :

Libre Office :
9 526 750 lines of code (C++)
1871 contributors since 2000
about 80 contributions/month

Jquery :
38 489 lines of code (Javascript)
369 contributors since 2006
about 10 contributions/month

Open Source Initiative – affiliate members
http://opensource.org/affiliates

Yes, we can

- Open source
  - Cooperation
    - Distributed
    - Continuous
    - Collective decision making
    - Multi-resources
      - Documentation
      - Design
      - Dev
      - Translation
      - ...

Open Source Initiative – affiliate members
http://opensource.org/affiliates

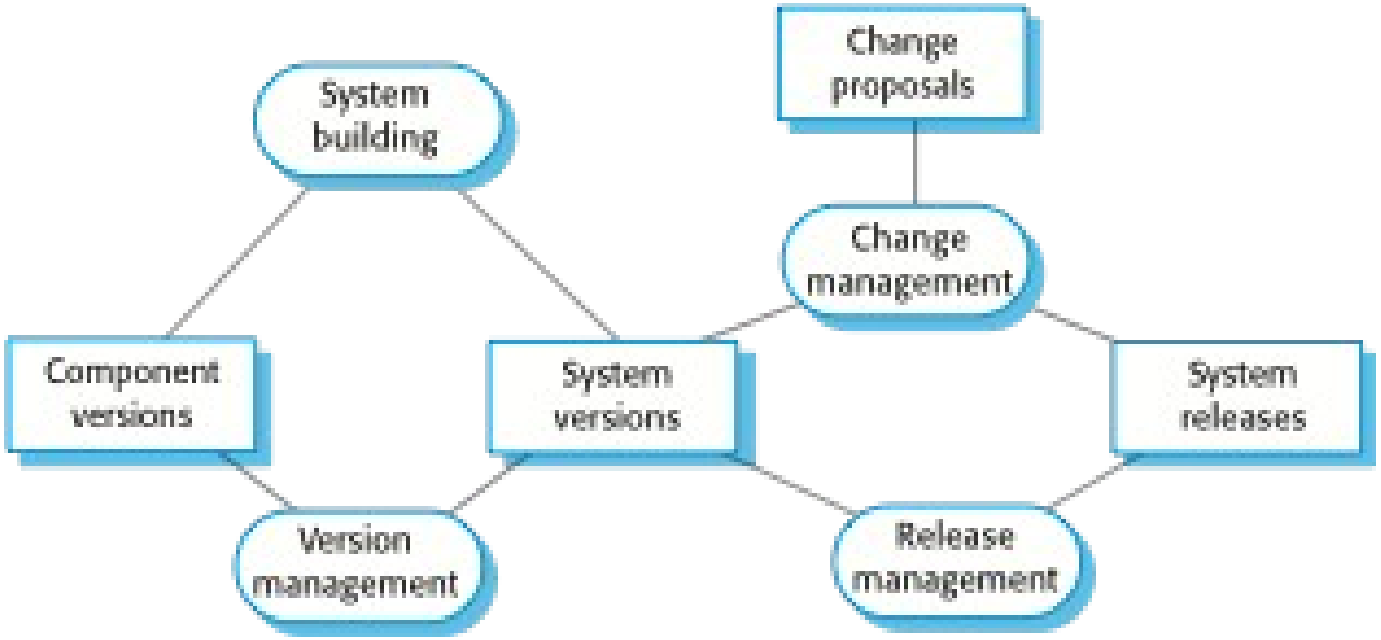
« Configuration? »

- Configuration management and control
  - An inspiration from system engineering
    - Process and tools that enable to sustain the soundness of a system with its specifications (functional and non-functional)
    - Handle changes in a systematic way to keep the system integrity (bugs and new features)
      - Identifying and recording the system states (baseline ➔ new version)
    - Quality assurance (QA) of the different system versions (Contrôle de la qualité)

➔ Software Configuration Management (SCM)

« Configuration? »

- The activities behind configuration management



« Configuration? »

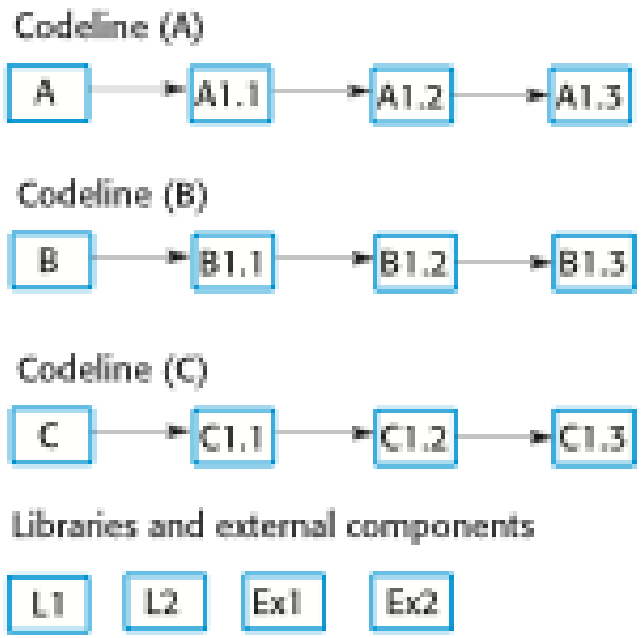
- Software Configuration Item (SCI – tr: Elément d'une configuration)
  - Any project resource that is kept under configuration control
    - Code
    - Design document
    - Test data set
    - User manual
    - Licence
    - […]
  - A collection of uniquely identified resources
  - That can exists in different version (history)

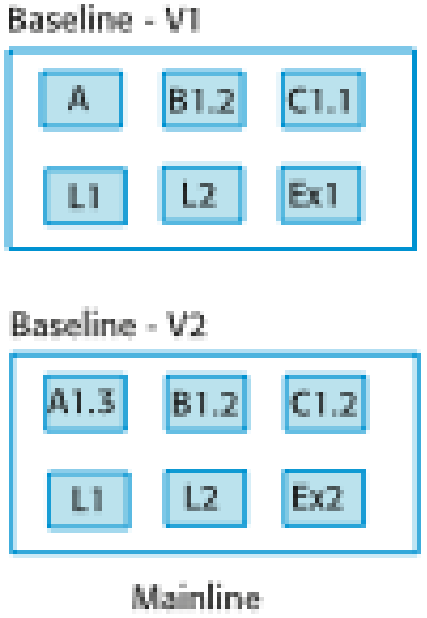
SCM and version management

- Configuration management for software engineering
  - Code version management system
  - Ex. : GIT, Mercurial, CVS, SVN…
  - Integrated in complete software project management solution (e.g., GitHub, SourceForge, JIRA/BitBucket …)
    - Bug and change tracking (Suivi de bug/évolution) --> Bugzilla, Trac
    - Cooperative editing solution --> Wiki
    - Communication tools --> Mailing-list, Chat/IRC, Slack...

Managing SCI and versions

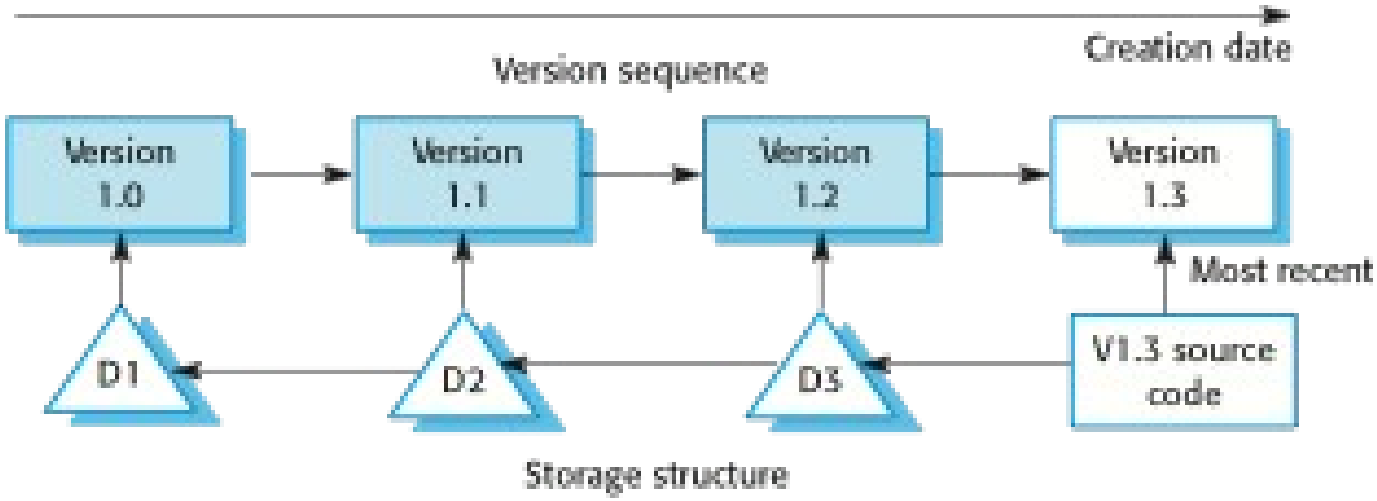
- The « production line » metaphor
  - Baseline : one or several reference version
  - Codeline : a specific production line (at a team or a developer scale)
  - Managing dependencies
    - Libraries
    - External components





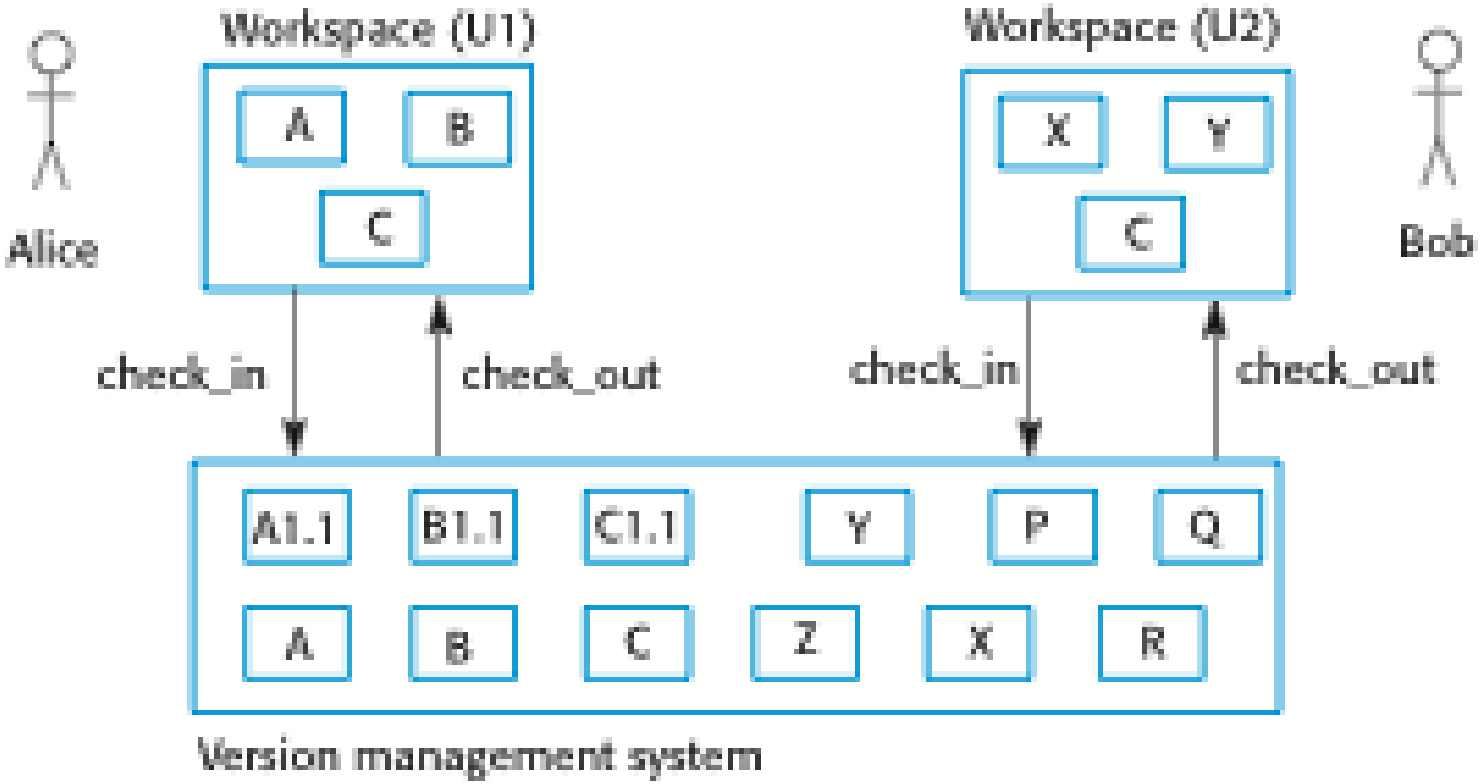
Version management

- Principles
  - Identify versions (ie, v 0.1, v 0.3, v 1.0 …)
    - At lower scale each revision (r122, r455, r1091…)
  - Keep track of the difference between versions
    - Store the last state of the current version and all the difference (the deltas) with the older one



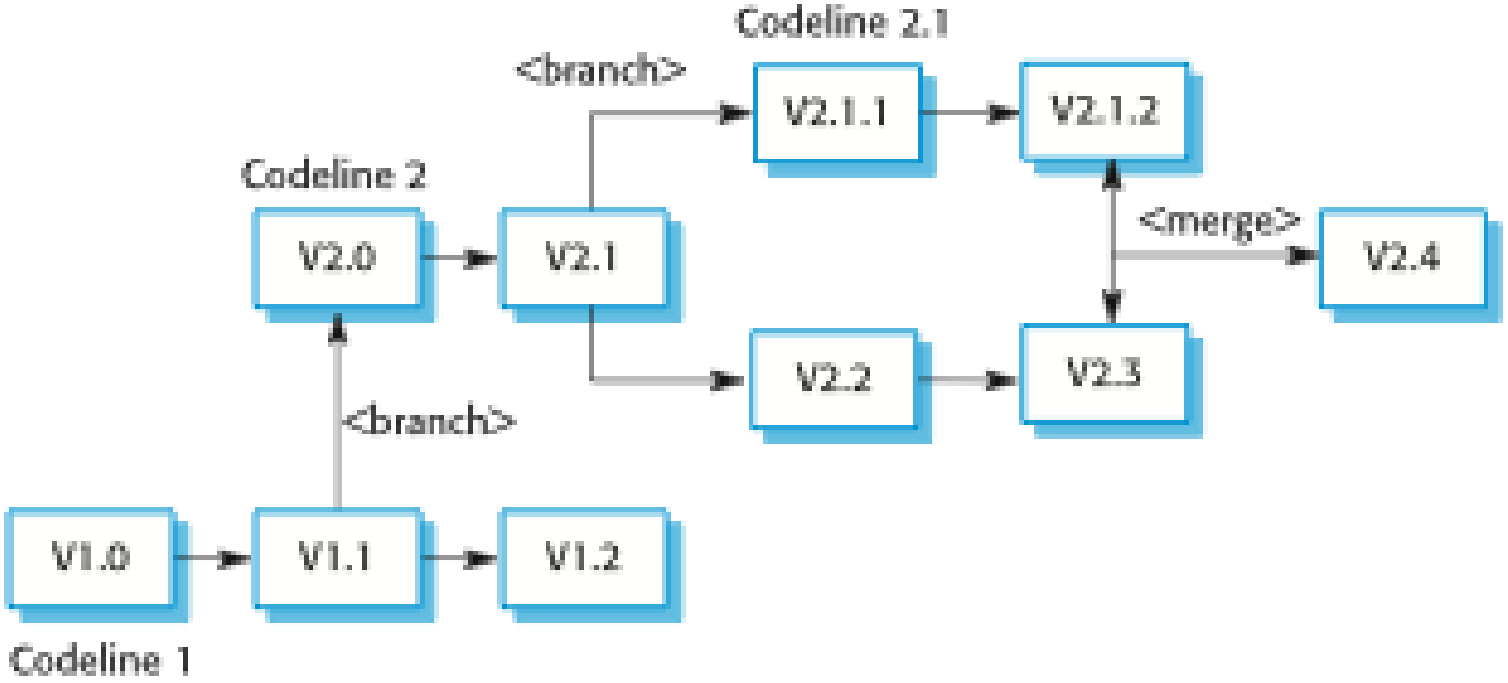
  - Centralized (CVS, SVN) vs Distributed (Git, Mercurial)

Working on a code repository




```
svn checkout <URI_dépôt_distant>
svn commit –m "change short desc"
```

Branch and merge




```
svn copy <URI_distant_repo/trunk>
<URI_distant_repo/branch>

svn merge <URI_distant_repo/branch>
<local_workspace_path/trunk>
```

Subversion concepts (SVN)

- The « project tree » - Three kind of code lines
  - « trunk » (Tronc)
    - Main code-line, the reference version for all contributors
  - « branches »
    - Paralell development
  - « tags »
    - Finalized stable version with a release identifier (*tag)*
      - ie, <URI_repo>/tags/mySoft-1.1
    - In fact, a tag is also a branch
      - But it is uncommon to work inside it!!!
      - Can evolve in special case (ie critical security bug correction)
- Useful

svn status (give the status of the local workspace against the repo – a.k.a. what will be for the next commit)
svn diff (displays the differences between revisions or specific files)

Subversion concepts (SVN)

- Conflict management

```
# following update of the local workspace
svn update
```

Subversion concepts (SVN)

- Conflict management

```
# following update of the local workspace
svn update
Updating '.':
Conflict discovered in 'foo.c'. Select: (p) postpone, (df) diff-
full, (e) edit, (mc) mine-conflict, (tc) theirs-conflict, (s) show
all options:
```

  - Postpone (handle manualy - « à la main »)
    - Mark the conflicting parts/lines in the file
    - Create temporary copies of both version (mine and their)

Subversion concepts (SVN)

- Conflict management

```
# following update of the local workspace
svn update
Updating '.':
Conflict discovered in 'foo.c'. Select: (p) postpone, (df) diff-
full, (e) edit, (mc) mine-conflict, (tc) theirs-conflict, (s) show
all options:
```

  - Pospone (Handle manualy - « à la main »)
    - Mark the conflicting parts/lines in the file
    - Create temporary copies of both version (mine and their)

➔ To do: Manual editing in order to define what will be the file at the end for the next commit

```
svn resolve --accept working <file>
```

Subversion concepts (SVN)

- Conflict management

```
# update du workspace local
svn update
Updating '.':
Conflict discovered in 'foo.c'. Select: (p) postpone, (df)
diff-full, (e) edit, (mc) mine-conflict, (tc) theirs-conflict,
(s) show all options:
```

  - diff-full : Display all the difference (btw local and repo)
  - edit : (default config editor) live editing of conflicts
  - mc : keep my local workspace as reference version
  - tc : overwrite the local workspace
  - show all : display all the possible options (mine-full, their-full)

➔ (tc or mc) at the end my local workspace will be commit to the repo (as a new reference version for trunk)

Good pratices

- At the end, no « branch » or « trunk » command…
  - trunk/branches/tags as a recommended organization
  - only directories
  - Convention: the last stable version trunk

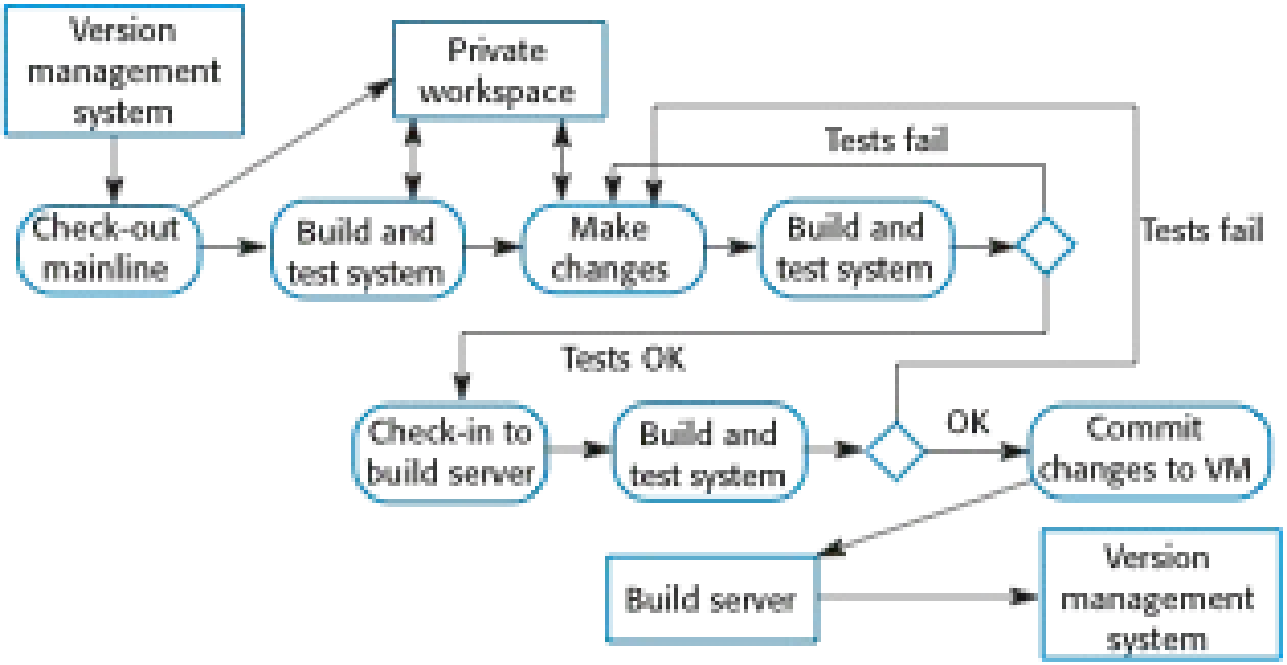
- Working with branches
  - One by person / by feature
  - Keep the branch updated with the trunk
  - Merge in « trunk » as a project team decision

Dealing with multiple versions

- Maintaining several version
  - Develop and compile for several technical environment : by operating system, by hardware, by client, …
  - Web and interoperability standard can limit the number of version…
  - … however still a current trend (ie. Smartphone environment)

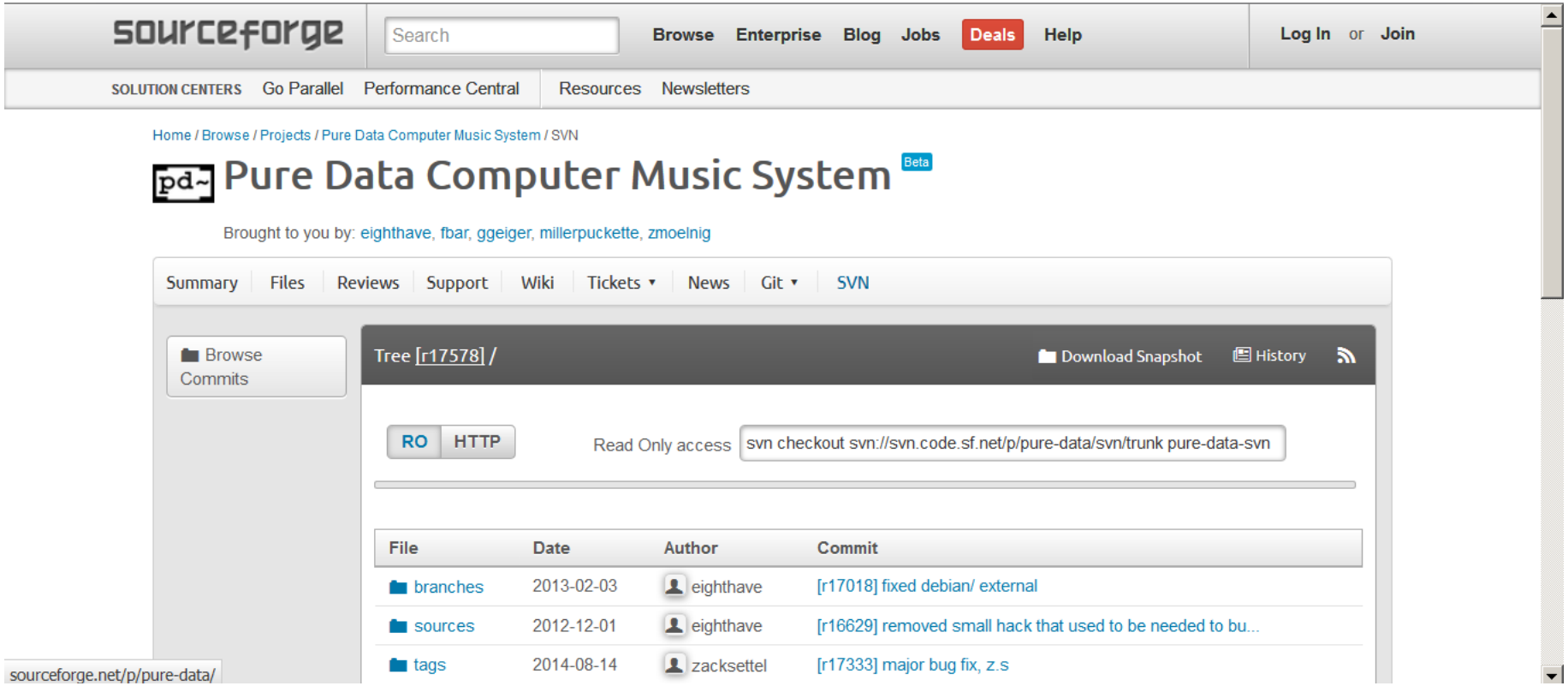
Continuous integration

- From development to deployment
  - An infrastructure for rapid delivery
  - --> How can bug correction/change be deployed fast after completion
  - Dev-ops : Development & Operation
- The continuous intergration model



For GL02

- Sourceforge (SF) & SVN
  - Each team will create a SF project with a SVN repository



Exemple de projet sourceforge, http://sourceforge.net/projects/pure-data/

References

- Thanks for your attention
  - Question(s) ?

- B. Collins-Sussman, B. Fitzpatrick & C. Pilato, *Version Control with Subversion*, Oreilly Media, http://svnbook.red-bean.com/
- K. Fogel, *Producing Open Source Software How to Run a Successful Free Software Project.* O'Reilly, 2013. http://producingoss.com/
- I. Sommerville, *Software Engineering*. Pearson Education, 2009.
- W. Scacchi, Understanding the requirements for developing open source software systems *Software, IEEE Proceedings* -, Vol. 149, No. 1. (2002), pp. 24-39

Annexes

- Pratiques des communauté source
  - Point sur les licences
  - Des différences de pratic
  - Exemples