Distributed Source Code Management tools

Itaapy working experience

Luis Belmar-Letelier

luis@itaapy.com

Itaapy

Content

- Leaving CVS
- Distributed source control (DSC)
- Arch/tla
- Git/Cogito

Leaving CVS in 2002: Context

- **What**
  - open source project `localizer`, `itools`
  - customer projects: quality agent process
  - full Unix environment
- **How**
  - XP: pair programming
  - XP: code review
  - XP: test driven code
  - ... more than two developers on each piece of code

Leaving CVS in 2002: Context

- **Using and making version control system**
  - Using version control to make software.
  - Making for our customers CMS applications with versionning
    - using Zope
    - using Plone
    - using CPS
    - using iKaaro

So ok let’s go out of CVS

Leaving CVS in 2002

Centralized Version Control?

Leaving CVS in 2002

Centralized Version Control?

No thanks

Leaving CVS in 2002

Centralized Version Control?

No thanks

Why do we need DSC?

- **To work offline, with no mass commit back from:**
  - the train, the sea, the beach, Montmartre
- **We don’t need to give write access to the archive**
  - useful for itools contributors
  - useful in customer projects (each developer is the only responsible for the state of his tree
  - lower infrastructure cost, no root access to set up a CVS repository, no +w on group, with umask funky config.

In 2002 we went to Arch/tla

In 2002 `Arch/tla` provide real Distributed Source Control:

- **Documentation**
  - 100 pages of real documentation
- **Tools**
  - tla
  - tla-tools
  - archzoom
  - tla-cvs-sync

Our first real-life experience with DSC.

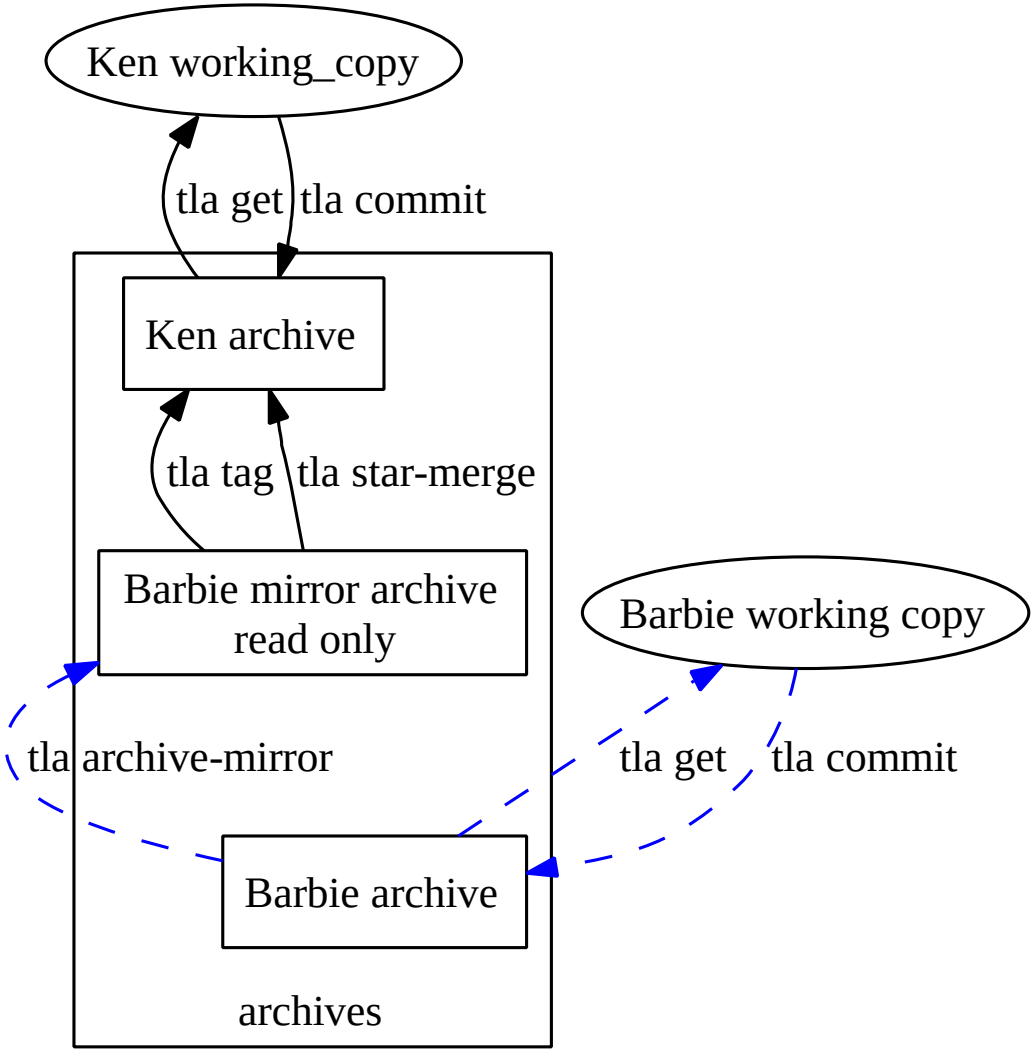
Two years after... in 2004

The target was reached

Big improvement of the QA process

- fine control of the business projects releases
- easying the simultaneous work of many developers at the same time on the same code.
- precise and reproductible state of each release of code.
  - fine control on the customer releases
  - each developer version of the code
  - old production server
  - pre-production server
  - customer preview server

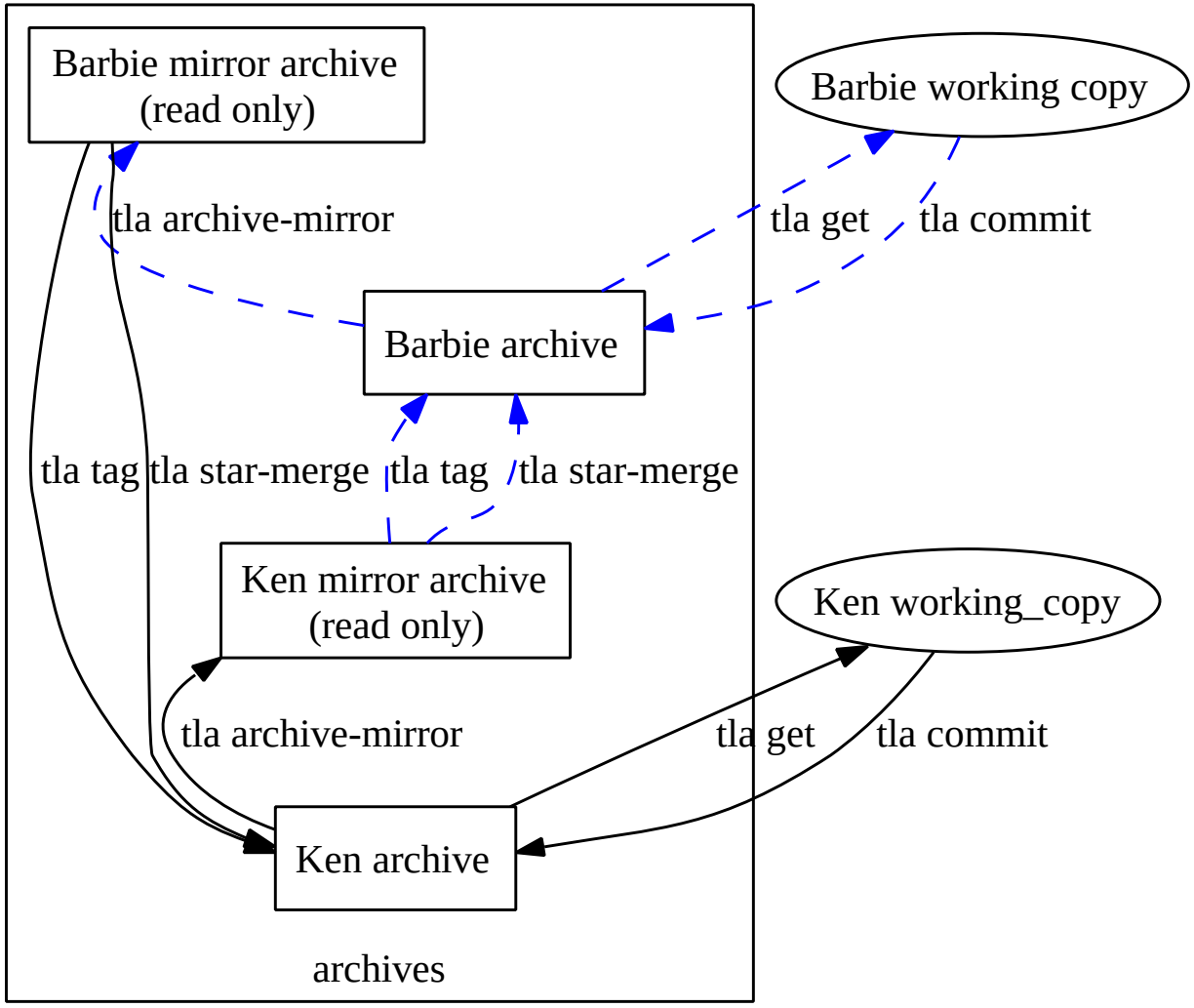
Arch/tla: Distributed working



Target reached but we left arch/tla

- learning it is hard
- mastering it is very long
- programming interface is hard
- persistence layer is slow
- improving it with caching adds complexity
- mirror based sharing is good for customer projects but mail patch submit should be better for Open Source project developpement.
- bad feadback from the itools community

Target reached but we left arch/tla



Target reached but we left arch/tla

The hiden idea ...

- itools goes to a VirtualFileSystem layer, the persistance layer given by Zope, then ZODB, then File System files
- `itools.cms` aka iKaaro implement versionning for hight level CMS objects.

In `itools` next releases we expect to use DSC for the direct persistant layer.
For this we need:

- Solid File Sysem backend
- speed, speed, speed

Candidates to replace arch/tla

Candidates to repleace `arch/tla`

- Monotone
  - the paranoid one (strong cryptography)
  - own network protocol
- Mercurial
  - written in Python
  - still beta at the time of choosing
- Codeville
  - best merge algorithm, own network protocol
- Darcs
  - stable and simple
  - scales not very well

Candidates to replace arch/tla

Candidates to repleace `arch/tla`

- Monotone
  - the paranoid one (strong cryptography)
  - own network protocol
- Mercurial
  - written in Python
  - still beta at the time of choosing
- Codeville
  - best merge algorithm, own network protocol
- Darcs
  - stable and simple
  - scales not very well

Candidates to replace arch/tla

Candidates "La finale"

- Bazaar-NG (bzr)
  - written in Python, supported by Canonical
  - still alpha at the time of choosing
- git/cogito
  - stable, many tools around, scales very well, strong community
  - best evidence: the Linux kernel

3 programming interfaces

- **Cogito**
  - Very low learning cost (10 min)
  - full power of Git with a set of 15 commands
  - natural set of commands
- **Git**
  - Controling the internal index structure
  - made in Bash, C, Perl, Python
  - and soon available as **C library**
- **The File System archive `.git/`**
  - The persistant layer is dam robust
  - it’s fast, fast, fast

Cogito A natural `cmd` Set

- `cg-add`, `cg-mv`, `cg-rm`, `cg-restore`
- `cg-status`, `cg-diff`, `cg-commit`, `cg-log`
- `cg-clone`, `cg-update`, `cg-merge`
- `cg-switch`

Git/Cogito tools

- Full migration tools SVN, CSV, arch
- Web Navigation gitweb
- Rich-client revision tree visualizer: gitk, qgit

References

- This slides
  - http://doc.ikaaro.org/DSC_management.pdf
- A day to day use of git/cogito documentation is available on the itools documentation, (11 pages).
  - http://doc.ikaaro.org/itools-doc.pdf
  - chapter "Keeping track of `itools` with *Git/Cogito*"
  - chapter "Contributing to `itools` with *Bugzilla* and *Git/Cogito*"
- The itools mailling list
  - **itools@ikaaro.org**