To Distribute or Not To Distribute

How to know your DVCS from your CVCS

Jim Hague[1] Russel Winder[2]

[1]Ifield Computer Consultancy
(not Canonical Ltd.)

[2]Concertant LLP
(not Canonical Ltd.)

ACCU 2008

Copyright © 2008 Jim Hague, Russel Winder

Aims of the Session

- Introduce the distributed and centralized models of version control.
- Convince people that distributed version control systems are superior in all ways to centralized version control systems.

Copyright © 2008 Jim Hague, Russel Winder

Structure of the Session

- Beginning.
- Middle.
- End – possibly.

Copyright © 2008 Jim Hague, Russel Winder

Begin

- Why use version control?
  - That's another talk...

Copyright © 2008 Jim Hague, Russel Winder

Begin Again

- Where are we, and how did we get here?
- Some contemporary version control systems.
- One or two systems in bit more depth.
- How does distributed version control work?
- Version control workflows.
- Opinion and prejudice unleashed: comments on systems we have used.

Copyright © 2008 Jim Hague, Russel Winder

Prehistory

- Card decks and paper tape store programs.
- Evolving code is hard enough without worrying about versions.
- Each program has maybe one client, no need for a publishing strategy.

Copyright © 2008 Jim Hague, Russel Winder

The Dark Ages

- Programs stored on disk or magnetic tape.
- Backups are taken – but this is for the operators' benefit.
- Evolving code is hard enough without worrying about versions.
- Each program has maybe one client, no need for a publishing strategy.

Copyright © 2008 Jim Hague, Russel Winder

The Middle Ages

- Programmers make errors and want to revert back to the old version...
- ...and thus did SCCS spring into being.

Copyright © 2008 Jim Hague, Russel Winder

SCCS and RCS

- Individual files are versioned.
- Entirely file system based.
- Network use problematic.
- Lock – edit – (commit –) unlock cycle.

Copyright © 2008 Jim Hague, Russel Winder

The Renaissance

- Concurrent edit and merge model of working.
- Client–server access.

Copyright © 2008 Jim Hague, Russel Winder

The Enlightenment

- Changesets
- Atomic commits

Copyright © 2008 Jim Hague, Russel Winder

The War Years

- Perforce in profit making organizations.
- CVS for FOSS.
  - CVS has too many problems...
  - ...enter Subversion.

Copyright © 2008 Jim Hague, Russel Winder

The Jet Age

- The rise of peer-to-peer.
- Operating without network connectivity – yes, it does happen!
- The fascism of centralization.

Copyright © 2008 Jim Hague, Russel Winder

The Second Renaissance

- Arch, TLA.
- BitKeeper and the Linux kernel.
- Monotone and Git.
- Bazaar and Mercurial.
- Darcs.

Copyright © 2008 Jim Hague, Russel Winder

Some Contemporary Version Control Systems

Centralized

- Subversion
- Perforce
- ClearCase
- Vesta

Distributed

- Bazaar
- Mercurial
- Git
- Darcs
- Aegis

Copyright © 2008 Jim Hague, Russel Winder

Two DVCS Systems In Action

- Bazaar In Action
- Mercurial in Action

Copyright © 2008 Jim Hague, Russel Winder

Distributed Version Control

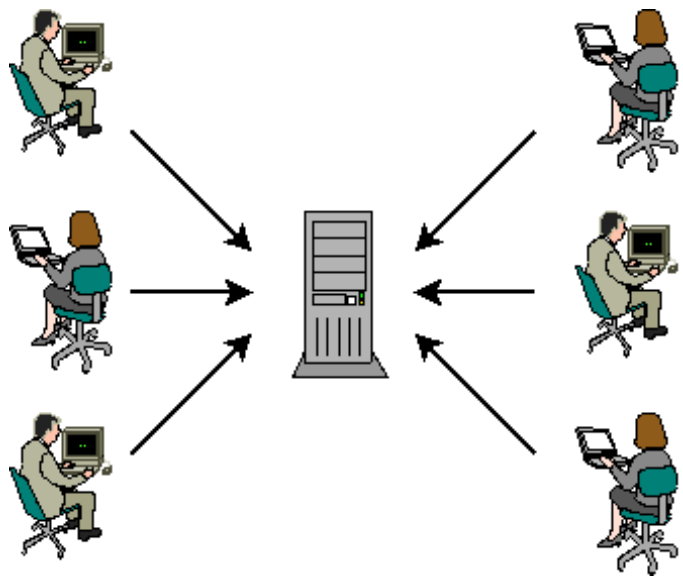
How being distributed works

- Centralized Version Control is easy(ish) to comprehend:
  - Linear series of changes.
  - Each change has a context.
- In DVCS, every developer has their own branch(es).
- You can merge between any branch.
- You're going to need some decent merging then...

Copyright © 2008 Jim Hague, Russel Winder

Workflow – Centralized

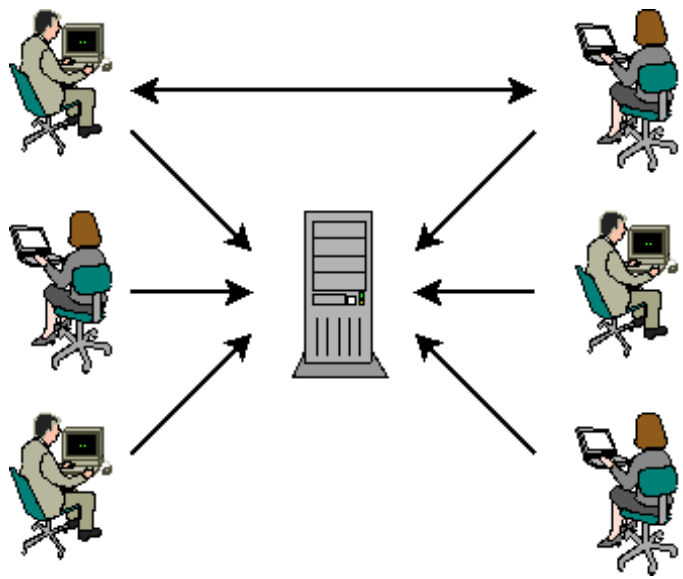
- Update/Edit/Update-Merge/Commit.
- Branch on server/Update checkout to branch.



Copyright © 2008 Jim Hague, Russel Winder

Workflow – Distributed as Centralized

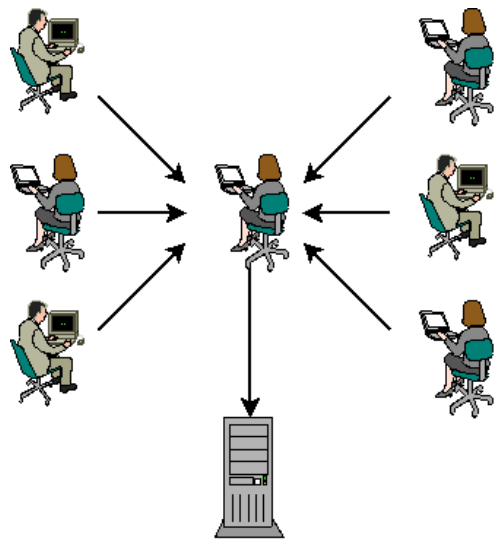
- Pull/Update/Edit/Commit/Pull-Merge/Commit/Push.
- Branch on server/Pull branch/etc.



Copyright © 2008 Jim Hague, Russel Winder

Workflow – Distributed, Centralist with Review

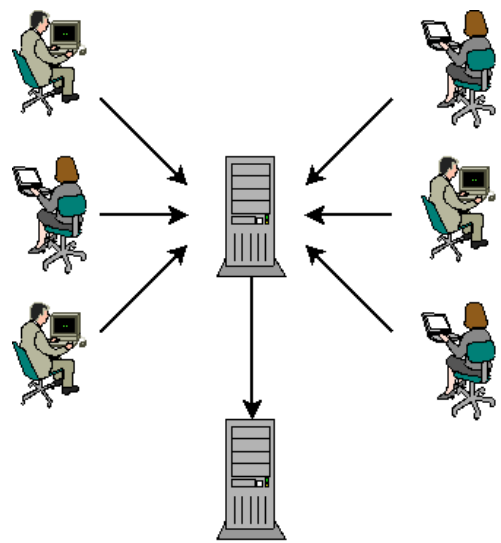
- Pull/Update/Edit/Commit/Pull-Merge. Inform reviewer who pulls, or email change bundle to reviewer. If review passed, reviewer commits in reviewed tree and pushes to master. Or authorizes direct push to master. Or something.



Copyright © 2008 Jim Hague, Russel Winder

Workflow – Distributed, Centralist with Review

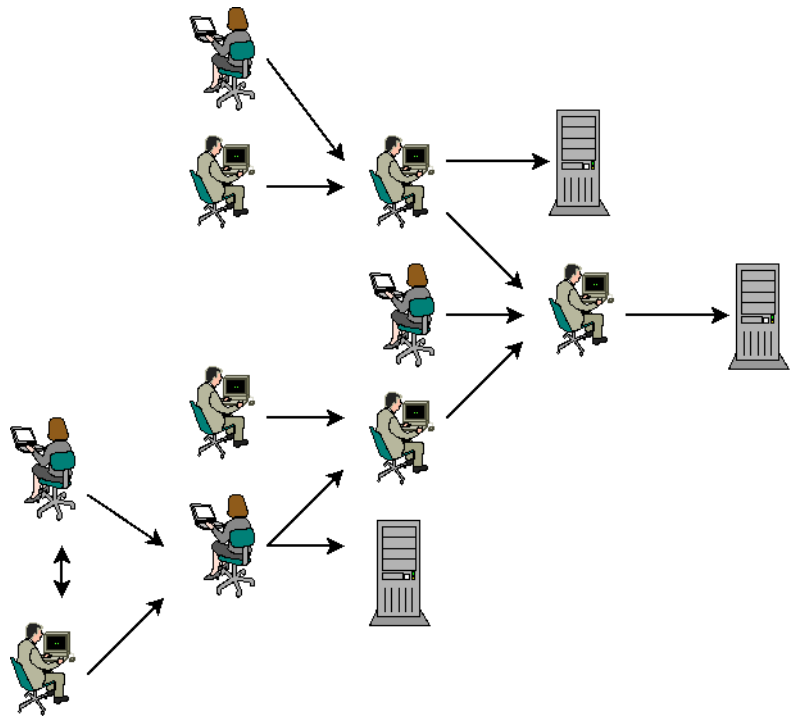
- The review gateway can be an automated process (e.g. Bazaar's PQM).
- Pull changes from a branch, make sure everything still builds and the test suite passes.
- Commit the changes and push to the master.



Copyright © 2008 Jim Hague, Russel Winder

Workflow – Distributed, Disconnected Hierarchy

- Reviewers don't scale well.
- Answer – a tree of reviewers!



Copyright © 2008 Jim Hague, Russel Winder

Opinions – CVS

- In its time is was superb.
- Now a millstone – time has moved on.

Copyright © 2008 Jim Hague, Russel Winder

Opinions – Subversion

- Excellent – but only if everyone using a repository is connected to the repository at all times.
- Well actually will be excellent by v1.6, currently only good.
- Well, to be honest, totally useless for any form of distributed/virtual organization.

Copyright © 2008 Jim Hague, Russel Winder

Opinions – Perforce

- Payware – $800, then $160 per seat per anum.
- Long established and innovative.
- Highly portable.
- Fast.
- Excellent merge capabilities including merge tracking.
- Workspace can be loaded from multiple sources in repo.
- Work in progress tracked on server.
- Proxy servers, and access to remote servers.
- Nice centralized system. But we want distributed...

Copyright © 2008 Jim Hague, Russel Winder

Opinions – Bazaar

- Simple and usable.
- It works.
- High-profile backers – e.g. Canonical.
- Eclipse Integration.
- Works well as a front end to Subversion as well as being an excellent centralist of distributed version control system.
- Compared to Git and Mercurial, Bazaar is slow.

Copyright © 2008 Jim Hague, Russel Winder

Opinions – Mercurial

- Simple and usable.
- It works.
- Fast (includes a few C modules for speed).
- High-profile users, e.g. Sun.
- Integrated 'MQ'quilt-like system.
- Has a Good Book.
- TortoiseHg Windows integration.
- NetBeans integration.

Copyright © 2008 Jim Hague, Russel Winder

Opinions – Git

- By Linux hackers for Linux hackers.
- Complex but very flexible.
- Plumbing and Porcelain.
- Great for working with Subversion repositories!

Copyright © 2008 Jim Hague, Russel Winder

Opinions – SVK

- OK if what you want is disconnected working with a Subversion repository.
- It's in Perl...
- Actually Bazaar and Git are better.
- Except that SVK is a Subversion store whilst Bazaar and Git are not.
- Cannot be used in a fully-distributed way.

Copyright © 2008 Jim Hague, Russel Winder

Opinions – ClearCase

- Reminder – we've still never used ClearCase.
  - We are still going to express an opinion even though we have only done some reading.
  - If you have real knowledge, please share.
- Data repository holds:
  - Current and historical source.
  - 'Derived objects'.
  - Accounting data - who created what, why? What is in a build? etc.
- Each source tree or subtree can be a separate VOB.
- VOBs distributed over LAN, can be linked into single logical tree.
- User access via *views*. VOBs appear as trees in views.
- Work on own branch.
- Good merging.

Copyright © 2008 Jim Hague, Russel Winder

Opinions – ClearCase

- ClearCase MultiSite
  - Site gets a branch writable by that site only.
  - All other branches visible, but read-only.
  - Off-network working not possible.
- No atomic commits.
- Expensive, in resources in $$$.
- BigCo stuff.
- From our point of view, not a true DVCS.

Copyright © 2008 Jim Hague, Russel Winder

Opinions – Microsoft Foundation Team Server

- Reminder – we've never used it.
- Team Foundation Version Control.
- Changeset based.
- Looks similar to ClearCase.
- Off-network working not possible (as far as we can tell).
- From our point of view, not a true DVCS.

Copyright © 2008 Jim Hague, Russel Winder

Rounding Off

- Centralized systems such as Subversion are yesterday's version control systems.
- Distributed systems such as Bazaar, Mercurial and Git are today's and tomorrow's systems.

Copyright © 2008 Jim Hague, Russel Winder

How Mercurial Works

Introduction

- `http://hgbook.red-bean.com/hgbookch4.html`, `http://www.selenic.com/mercurial`
- Revlog – thing that holds a file and its changes
- NodeId – SHA1 hash of current file contents and NodeId(s) of parent(s)
- So, NodeIds match iff file states are identical and the file has the same history

Copyright © 2008 Jim Hague, Russel Winder

How Mercurial Works

The Manifest

- List of all files in the project with their current NodeIds.
- Manifest is held in a revlog.
- Manifest NodeId identifies project filesystem at a particular point.

Copyright © 2008 Jim Hague, Russel Winder

How Mercurial Works

The Changeset

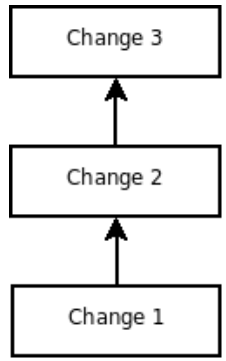
- Atomic collection of changes to a repository leading to new revision.
- Manifest NodeId.
- Timestamp and committer ID.
- List of changed files.
- NodeId of parent changeset(s).
- Changelog is a revlog of the changesets.

Copyright © 2008 Jim Hague, Russel Winder

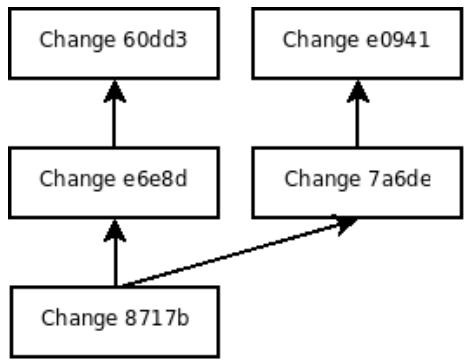
How Mercurial Works

Pushing and pulling

- Copy changes from one repository to another



- Find common ancestor
- Create parallel track/head/branch from common ancestor

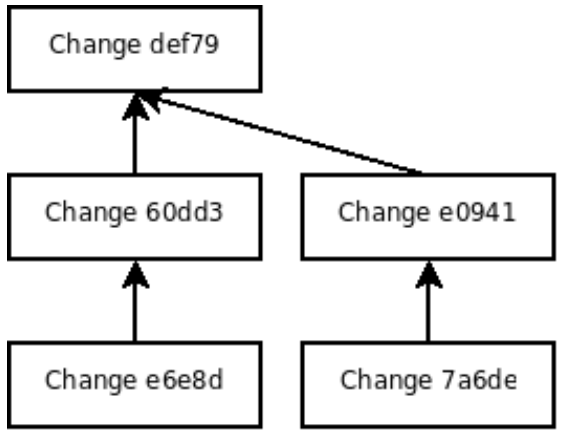


Copyright © 2008 Jim Hague, Russel Winder

How Mercurial Works

Merging

- Merge two heads
- Merge creates a change
- Human resolves conflicts
- Committing change collapses two heads into one



Copyright © 2008 Jim Hague, Russel Winder