Version Control Systems

Petr Baudis
pasky@suse.cz
SuSE Labs CZ

Novell

Let's talk about...

What are we talking about (version control basics)

How various systems differ (version systems taxonomy)

Where are we now (current systems overview)

What are we working on (possible future trends)

© December 11, 2003 Novell Inc.

What are we trying to do?

Preserve development history

- Browse it
- Return to previous versions

Enable collaboration

- Multiple people can work at once
- Access to the bleeding-edge for users

Manage parallel development

- Branching and merging

© December 11, 2003 Novell Inc.

Revisions

revision =~ commit =~ changeset

Basic unit of the history

State of the project (file) at a particular moment

Has some identificator and description

Either per-file or for the whole tree

© December 11, 2003 Novell Inc.

Branches

branch =~ line of development =~ head

Project can have multiple “latest versions”

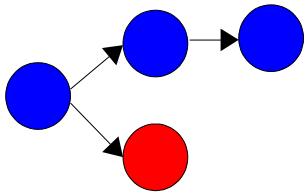
- Development vs. stable branches
- Experimental development efforts
- Personal staging area

Branches are easy to fork, harder to merge

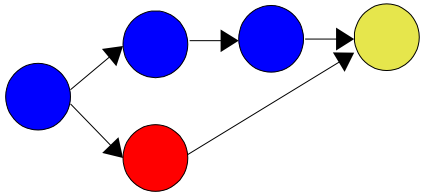
© December 11, 2003 Novell Inc.

Revision Graph

Revisions in separate branches form a tree:



Some systems can express merges in the graph, making it a DAG:



© December 11, 2003 Novell Inc.

Novell

Repositories

Database for all this

One of the most variant aspects

- Per-file history in custom format
- Tarballs with revisions
- SQL database

Need to access somehow

- Dedicated server
- Dumb transport
- Special access methods (SSH, WebDAV)

© December 11, 2003 Novell Inc.

Novell

Version Control Taxonomy

Development Model

Centralized development

- Traditional way of doing it
- Single canonical repository
- Most information not available locally

Distributed development

- Anyone can clone a repository and commit locally
- Usually, you have the history locally too
- More freedom for the developer

© December 11, 2003 Novell Inc.

Distributed Development

Pros:

- Independent parallel development
- Version control for your private forks
- Version control for your offline work

Cons:

- More complicated
- Can be confusing at first

© December 11, 2003 Novell Inc.

Distributing Your Work

- **Push** to a remote repository
- Let others **pull** from your repository
- Submit patches

Usually a central repository:

- Synchronization point for developers
- Following official development by users
- Commit policy either:
  - Developers push at will (CVS-like)
  - Project lead pulls from regular mortals and publishes that as the central repository (Linus)

© December 11, 2003 Novell Inc.

Identifying Revisions

Numerical (1.234, 34568)

- Non-unique and non-stable in distributed systems
- External (if the revision is modified, ID stays the same)
- Human-friendly

Hash-based (46c9c209f359d2c72df9b1de2442d06a5208cb1a)

- Unique and stable in distributed systems
- Intrinsic (modified revision == different ID, where the "==" is cryptographically strong)
- Human-unfriendly (at least at first)

Symbolic (fix-evil-bug)

- Non-unique
- External, human-friendly but it depends

© December 11, 2003 Novell Inc.

Primary graph objects

Vertices

- Primary objects are states (snapshots)
- Commit: make a new snapshot
- Edges are just references to previous vertices
- GIT, CVS

Edges

- Primary objects are changes (patches)
- Commit: add the new change to the current set of changes
- Vertices are just sets of changes
- Darcs, GNU Arch

Rarely clear-cut; systems with vertices still frequently rely on edges for storage (delta optimization)

© December 11, 2003 Novell Inc.

Current Systems Overview

But we have CVS! ... right?

CVS is not the answer, CVS is the question.
No is the answer.

-- Theodore Ts'o

© December 11, 2003 Novell Inc.

Novell

Why the fuss?

Hey, but CVS is fine...

- Non-atomic commits
- Bad support for branches
- Cannot delete directories, rename files...
- Inefficient network communication

- And no distributed development

© December 11, 2003 Novell Inc.

Existing Tools

- Subversion
- Bitkeeper
- Darcs
- Monotone
- GIT/Cogito
- GNU Arch, Codeville, Mercurial, SVK, ...

Research vs. usability

© December 11, 2003 Novell Inc.

Subversion (SVN)

"CVS 2.0"

Quite stabilized and spreading widely

Cures most CVS design mistakes

Centralized and bad in merging

© December 11, 2003 Novell Inc.

Novell

Bitkeeper (bk)

The first popular distributed VCS

Proprietary and evil ;-)

Per-file history with revisions bundled to “changesets”

One repository per branch

Allegedly very good in merging

Large and somewhat arcane command set

- Interesting GUI tools

© December 11, 2003 Novell Inc.

Darcs

One of the two real-world OSS projects in Haskell

Quite exotic design

Focuses solely on the edges – at any point, you have a combination of patches initially based on empty tree

Very appealing, but does not scale well

© December 11, 2003 Novell Inc.

Monotone

The “Monotone design school” - major influence on several other systems

Focus on cryptographically strong accountability and consistency checks

Very flexible thanks to Lua scripting

Very slow, mediocre UI

© December 11, 2003 Novell Inc.

Monotone Object Model

Used in Monotone, GIT, Mercurial, Codeville

Object database, objects identified by SHA1 hashes

© December 11, 2003 Novell Inc.

GIT

Goddamn Idiotic Truckload of sh*t

Heavily influenced by Monotone design

Simpler, faster, (sometimes) more elegant

Focus on speedy handling of large trees

“Directory cache” for fast manipulation of working copy

Objects stored per-file, but can be combined to very efficient “pack files”

UNIX tool – collection of small focused commands

Usage ranging from reasonable to hopelessly complicated

- Cogito – frontend for GIT with strong focus on seamless UI and painless+intuitive usage

Still not stabilized, currently not very portable

© December 11, 2003 Novell Inc.

Future Trends

Enhancing Our Own

Large competition

There are *still* people starting new ones

There are *still* people willing to use those new ones (hopefully not so many of them anymore)

Plenty of duplicated effort

Things aren't as bad as they could be

- Vivid ideas communication (**http://revctrl.org/**)
- Reusing algorithms (eg. merging) and tools (gitk, gitweb)
- Interoperation between VCSes (Tailor, custom gateways)

© December 11, 2003 Novell Inc.

Novell

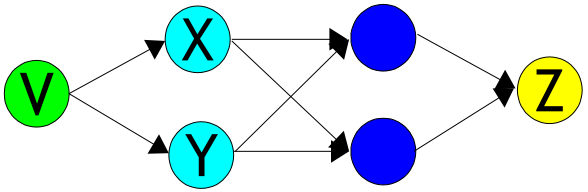
Merging Problem

Currently usually the three-way merge

- Take base B (LCA in rev. graph) and to-be-merged A and C
- Get diffs B->A and B->C and combine them
- Where they interfere, generate a conflict

Can generate way too many conflicts

Can silently get it wrong – e.g. the criss-cross merge:



What base for Z? X or Y?

The answer is V, but that has different problems

© December 11, 2003 Novell Inc.

Weaves

Comes from SCCS, used in BK

- Probably significantly contributed to BK's good merging
- Likely to get more widespread with PCDV merge

Weave (or “interleaved deltas”)

- Normally, file history is stored as base revision(s), other revisions are described by deltas against the bases
- Weave stores per-line history instead – whether a line was unborn yet, present or deleted, for each line and revision
- Makes certain operations (“annotate”) very fast, but is generally more complicated
- Does not blend well with Monotone-like systems
- Consequently, not so backup-friendly and harder to use with dumb servers

© December 11, 2003 Novell Inc.

Precise CDV Merge

Initially developed for Codeville (still in development)

Algorithm works just on the merged revisions, no base sel.

- Turn “on” any lines added in either revision
- Turn “off” any lines deleted in either revision
- Conflicts = on/off from both revisions between unchanged lines

Can be used for per-line changes as well as per-file changes (i.e. merging file renames) (but other methods are probably better for per-file changes)

Unfortunately a lot of border cases to solve

Still should work much better than three-way merge

© December 11, 2003 Novell Inc.

Cherrypicking Problem

Cherrypicking: When merging a branch, be able to choose only certain changes, or exclude some changes

Monotone-like systems don't work well at all here

Weave properties might make cherrypicking easy

© December 11, 2003 Novell Inc.

Novell

Homework Problem

Professor wants to know your solution, built up properly - not how you came to it

When forking a branch, manage it as a set of patches (also easy to cherrypick) which can be further updated

“Quilt” works best; attempts to integrate with VCSes

- Stacked GIT (StGIT), Mercurial extension (MQ)
- Also partial cherrypicking solution

© December 11, 2003 Novell Inc.

User Interface

Most systems are too hard/clumsy to use

Frequently ignored or dismissed

More regard to this lately

Graphical tools painfully missing

© December 11, 2003 Novell Inc.

...?

Questions?

Merging detailed:
http://revctrl.org/

© December 11, 2003 Novell Inc.

Novell

LUNCH NOW!

Thank you.

© December 11, 2003 Novell Inc.

Novell

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. Novell, Inc., makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All Novell marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of Novell, Inc. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

SUSE
A NOVELL COMPANY

Novell