Version Control Systems

Stefan Otte
Freie Universität Berlin

Proseminar Technische Informatik
Outline

Motivation

Centralized VCS
  CVS
  Subversion

Distributed VCS
  How does it work
  git

Conclusion

Optional
Why VCSs?

Why is it useful the software development process?

- Simplifies working in teams
Why VCSs?

Why is it useful the software development process?

- Simplifies working in teams **TOGETHER!**
Why VCSs?

Why is it useful the software development process?

- Simplifies working in teams **TOGETHER!**
- Track progress of work
Why VCSs?

Why is it useful the software development process?

- Simplifies working in teams **TOGETHER!**
- Track progress of work
- Jump back to older versions
Why VCSs?

Why is it useful the software development process?

- Simplifies working in teams **TOGETHER!**
- Track progress of work
- Jump back to older versions
- Handle concurrency
Why VCSs?

Why is it useful the software development process?

- Simplifies working in teams TOGETHER!
- Track progress of work
- Jump back to older versions
- Handle concurrency
Why VCSs?

Why is it useful the software development process?

- Simplifies working in teams **TOGETHER!**
- Track progress of work
- Jump back to older versions
- Handle concurrency

⇒ leads to easier and accelerated development
Short list of VCSs:

- CVS, AccuRev, Aegis, Arch, Bazaar, BitKeeper, ClearCase, CM+, CMSynergy, Co-Op, Darcs, Git, LibreSource Synchronizer, Mercurial, Monotone, OpenCM, Perforce, PureCM, SourceAnywhere, Subversion, Superversion, Surround, svk, Team Foundation Server, Vesta, and Visual SourceSafe
Version Control Systems

Short list of VCSs:

- CVS
- Git
- Subversion
The Centralized Approach

The Basic Terms:

- One central repository
- User retrieves working copy by doing a checkout
- can update his working copy
- a revision identifies a version of a file
- commit changes to the repository
- trunk is the mainline of dev.
- branch independent line of dev.
- tag certain revision

Central Repository
The Centralized Approach

The Basic Terms:

- One **central repository**
The Centralized Approach

The Basic Terms:

- One **central repository**
- User retrieves **working copy** by doing a **checkout**

Central Repository

FU Berlin, VCS, otte@inf.fu-berlin.de
The Centralized Approach

The Basic Terms:

► One **central repository**

► User retrieves **working copy** by doing a **checkout**

► can **update** his working copy
The Centralized Approach

The Basic Terms:

- One **central repository**
- User retrieves **working copy** by doing a **checkout**
- can **update** his working copy
- a **revision** identifies a version of a file
The Centralized Approach

The Basic Terms:

- One **central repository**
- User retrieves **working copy** by doing a **checkout**
- can **update** his working copy
- a **revision** identifies a version of a file
- **commit** changes to the repository
The Centralized Approach

The Basic Terms:

- One **central repository**
- User retrieves **working copy** by doing a **checkout**
- can **update** his working copy
- a **revision** identifies a version of a file
- **commit** changes to the repository
- **trunk** is the mainline of dev.
The Centralized Approach

The Basic Terms:

- One **central repository**
- User retrieves **working copy** by doing a **checkout**
- can **update** his working copy
- a **revision** identifies a version of a file
- **commit** changes to the repository
- **trunk** is the mainline of dev.
- **branch** independent line of dev.
The Centralized Approach

The Basic Terms:

- One central repository
- User retrieves working copy by doing a checkout
- can update his working copy
- a revision identifies a version of a file
- commit changes to the repository
- trunk is the mainline of dev.
- branch independent line of dev.
- tag certain revision
CVS’s History:

- CVS is an extended RCS
- Initial release 1985 by Dick Grune
- Ported to C in 1989
- Gained popularity…
CVS’s History:

- CVS is an extended RCS
- Initial release 1985 by Dick Grune
- Ported to C in 1989
- Gained popularity...
- BUT has many shortcomings
“CVS and its semi-chaotic development model have become cornerstones of open-source.”

by Ben Collins-Sussman
“CVS and its semi-chaotic development model have become cornerstones of open-source.”

by Ben Collins-Sussman

What does it mean exactly?

- Sourceforge used CVS
- hosted 100 000 FOSS projects
Subversion

Subversion is the successor of CVS:
- CollabNet initiated SVN in 2000
- SVN is CVS without its flaws
Subversion

Subversion is the successor of CVS:
- CollabNet initiated SVN in 2000
- SVN is CVS without its flaws
- basically replaced CVS
Features

“SVN - CVS done right”
Features

“SVN - CVS done right”

▶ atomic commits
Features

“SVN - CVS done right”
- atomic commits
- cheap branching and tagging (realized by copy)
Features

“SVN - CVS done right”

- atomic commits
- cheap branching and tagging (realized by copy)
- renaming and moving supported
Features

“SVN - CVS done right”

- atomic commits
- cheap branching and tagging (realized by copy)
- renaming and moving supported
- binary files are supported
“SVN - CVS done right”

- atomic commits
- cheap branching and tagging (realized by copy)
- renaming and moving supported
- binary files are supported
- more disconnected operations
Features

“SVN - CVS done right”

- atomic commits
- cheap branching and tagging (realized by copy)
- renaming and moving supported
- binary files are supported
- more disconnected operations
- v1.5: merge tracking improved merging dramatically!
Centralized Workflow

Central Repository
Distributed: How Does It Work?

Centralized user

Distributed user with local repository
Advantages of local repositories:

- the complete repository including history on local computer
- all operations are offline
- branches are local
- independent of the network
Advantages of local repositories:
- the complete repository including history on local computer
- all operations are offline
- branches are local
- independent of the network
- FASTER!
Distributed Chaos
History of git

Git - the fast version control system

- git is distributed
- developed to manage linux source code
- started in April 2005
- initial version developed in 2 weeks by Linus Torvalds
- now maintained by Junio Hamano
History of git

Git - the fast version control system

- git is distributed
- developed to manage Linux source code
- started in April 2005
- initial version developed in 2 weeks by Linus Torvalds
- now maintained by Junio Hamano
- used by:
  - git, Samba, X.org, Qt, KDE, VLC...
Linus Torvalds

“I’m an egotistical bastard, and I name all my projects after myself. First Linux, now git.”

“I hate CVS with a passion.”
Repository Structure

Working-tree

Remote Repository

commit
update
checkout
Repository Structure

**Working-tree** → **Index** → **Local Repository**

- `add`
- `commit`
- `commit -a`
- `checkout`
Repository Structure

Working-tree → Index → Local Repository → Remote Repository

- add
- commit
- commit -a
- checkout
- pull
- push
- fetch
Git’s Object Model

- tag
- commit
- tree
- blob
- reference
Git’s Object Model

- tag
- commit
- tree
- blob

SHA1 as name
Git’s Object Model

Content of a file
+ Header
+ Compression

= blob
Git’s Object Model

= tree

+ List of blobs/trees
+ metadata
+ compression
Git’s Object Model

Pointer to tree
metadata
compression +

commit =
Git’s Object Model

tag =

Pointer to commit + compression
Movable lightweight pointer to commit
Git’s Object Model

- tag
- commit
- tree
- blob
- reference
Git’s Object Model

tag

commit

reference

tree

blob
How Does a Commit Work?
How Does a Commit Work?

V 1.6.2

commit

branch X

HEAD

blob

tree

tree

tree

blob

FU Berlin, VCS, otte@inf.fu-berlin.de
How Does a Commit Work?
How Does a Commit Work?

V 1.6.2

commit

tree

commit

tree

blob

blob

blob

tree

branch X

HEAD
How Does a Commit Work?

V 1.6.2

branch X

HEAD

commit

commit

tree

tree

unchanged

blob

blob

tree

tree

blob

blob
How Does a Commit Work?

Diagram:

- V 1.6.2
- branch X
- HEAD
- commit
- tree
- blob
- tree
- blob
- tree
- blob
- tree
- blob
- tree
- blob
- tree
- blob
- tree
- blob
Shared Repository Model

Shared Repository

User + Local Repository
Integration Manager Model

"Main" Repository

Public Repositories

Integrator

Users
Dictator Lieutenant Model

Dictator → Official Repository

Dictator

Lieutenant

User

User
Conclusion

Use VCS!
Use VCS!

- CVCSs simplify development process
Use VCS!

- CVCSs simplify development process
- DVCSs can do everything CVCSs can
Use VCS!
- CVCSs simplify development process
- DVCSs can do everything CVCSs can
- plus much more
Thank you!
Questions?
Links

- git-scm.org
- Google Tech Talks:
  - Linus Torvalds on git
    http://www.youtube.com/watch?v=4XpnKHJAok8
  - git
    http://www.youtube.com/watch?v=8dhZ9BXQgc4
- Mercurial
  http://www.selenic.com/mercurial/wiki/
- Bazaar
  http://bazaar-vcs.org/
Mercurial

- Similar to git
- but design goal: simplicity
- GTT:
  http://www.youtube.com/watch?v=JExtkqzEoHY
Bazaar

- Used by Ubuntu
- pretty flexible
- Workflow: http://bazaar-vcs.org/Workflows
Merge and Rebase

**git merge**

```
c1 → c2 → c3 → c4 → c6 → c7
```

**git rebase**

```
c1 → c2 → c3 → c4 → c6
```

 FU Berlin, VCS, otte@inf.fu-berlin.de