Welches **Versionskontrollsystem** sollte ich nutzen?

Subversion, Git, Mercurial und Co

#webmontag #paderborn

4. Februar 2013

Hi!

Warum sollte ich zuhören?

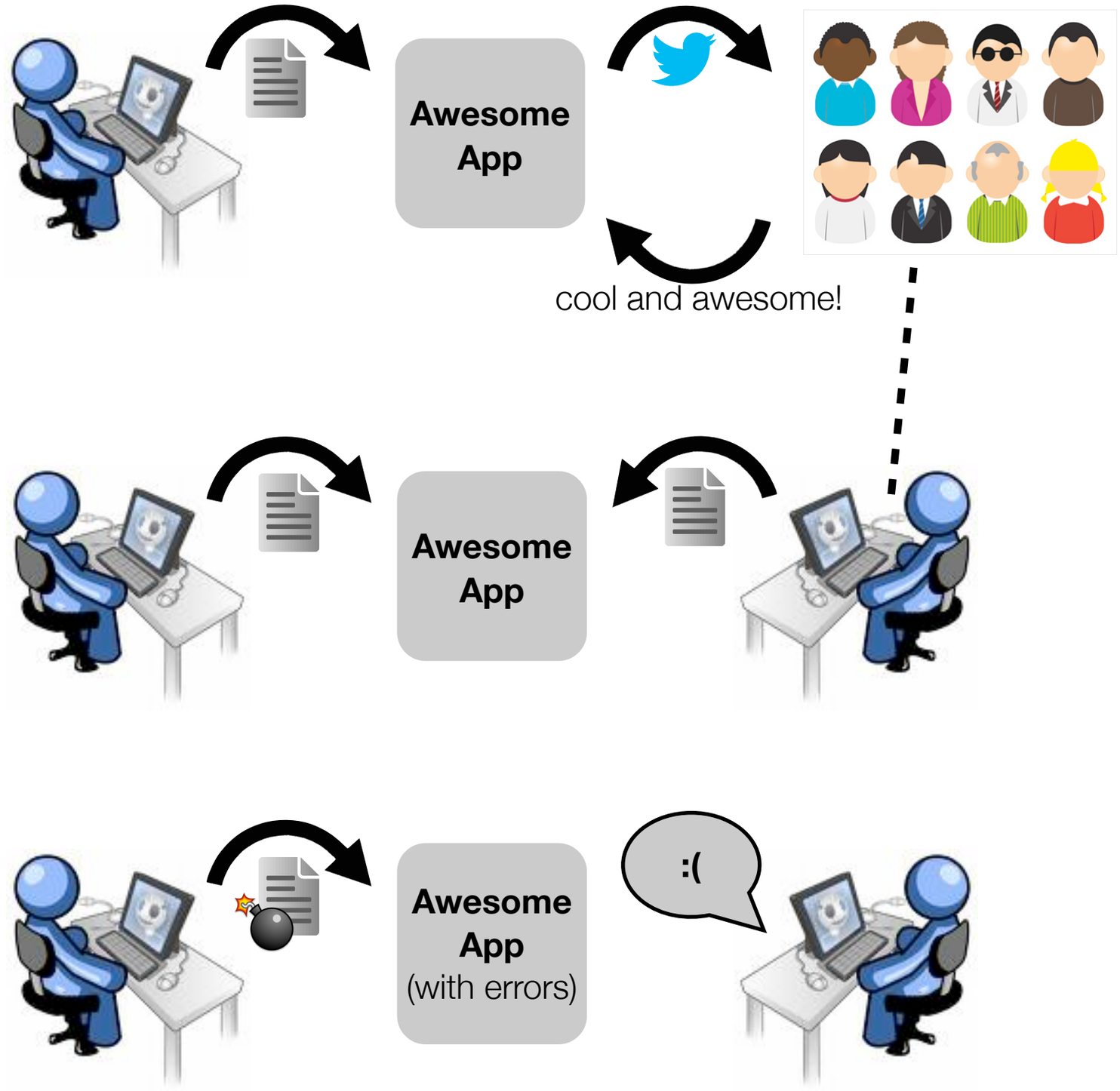
2012:

- Kundenprojekte
- iUPB
- PINGO
- private Projekte, Uni-Projekte
- Open-Source-Contributions

michael whittaker
webdesign. edv-dienstleistungen.

iUPB
powered by yippie.io

Warum ein VCS?



*„**Version control** is the art of managing changes to information“*

*„VC **system** is a general system that can be used to manage **any** collection of files“*

+ within a **team**

*For you, those files **might be** **source code**—for others, anything from grocery shopping lists to digital video [...] and beyond*

http://www.elixus.org/nightly/en/svk.intro.html

Lieblings-VCS: Welches nutzt du am meisten?

Tools

„A fool with a tool is still a fool“

VCS ist nur ein Tool

- Die besten Tools..
  - sind **unsichtbar**
  - **integrieren sich in deinen Prozess** und nicht umgekehrt

- VCS überhaupt notwendig?

- ausreichende „Versionierung“ schon vorhanden?
  z. B. **TimeMachine** oder **Dropbox**

- .NET/Visual Studio –> **Team Foundation Server** (Git-TF)

SUBVERSION

git

mercurial

2000

2005 ..... Zufall? 2005

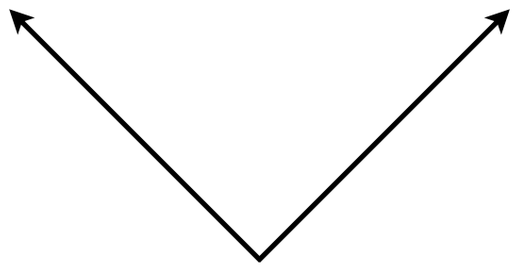
a better CVS

you can‘t do CVS right

zentral

verteilt

verteilt



sehr ähnlich

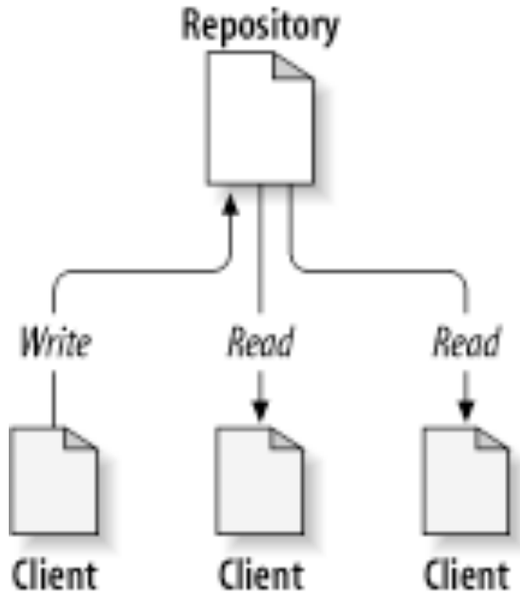
Kriterien

- Konzept und Workflow
- Setup und Verwaltung
- Kommando-Zeile, GUIs

Konzept

Konzept

- **zentrales** Repository
  - (Ordner, SVN-/SSH-/Web-Server)
- Benutzer/Client hat eigene Arbeitskopie
- nach Fertigstellung werden die Änderungen zum Repository übertragen (**commit**et).



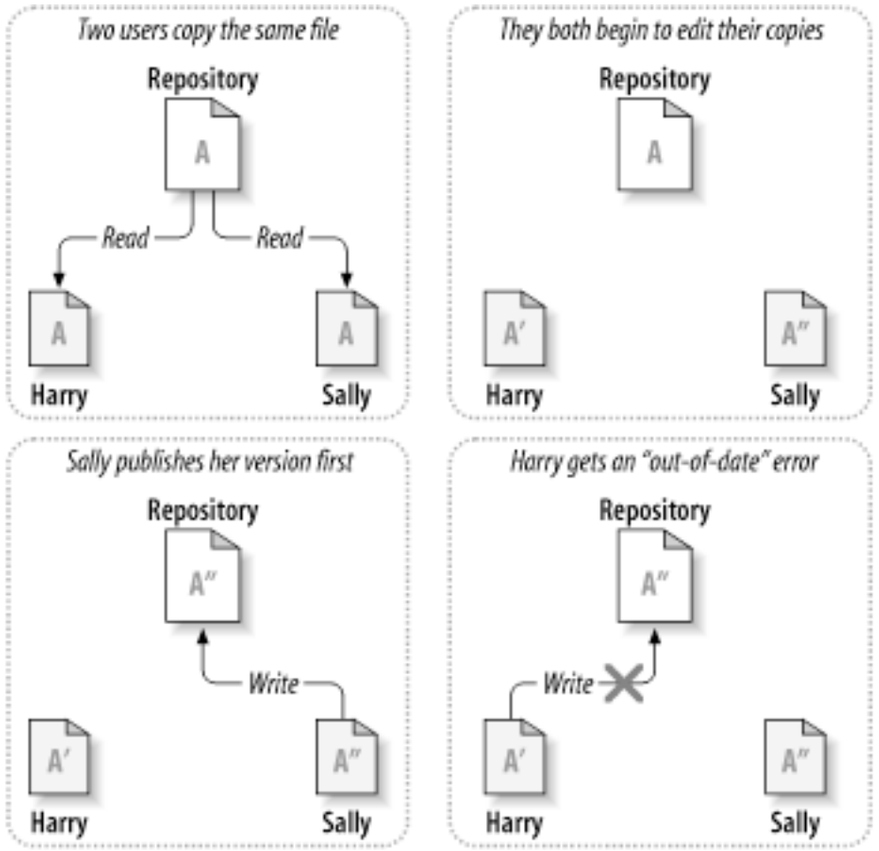
http://onlamp.com/pub/a/oreilly/opensource/news/subversion_ch02.html

Konzept: gleichzeitige Änderungen / Konflikte

- Benutzer arbeiten gleichzeitig an verschiedenen und gleichen Dateien

- Benuter „Sally“ ist fertig und commitet die Arbeit (= Senden an Server)

- Benutzer „Harry“ ist fertig, kann aber nicht commiten, da sein Stand nicht mehr aktuell ist



Vorbeugen:
Datei sperren?

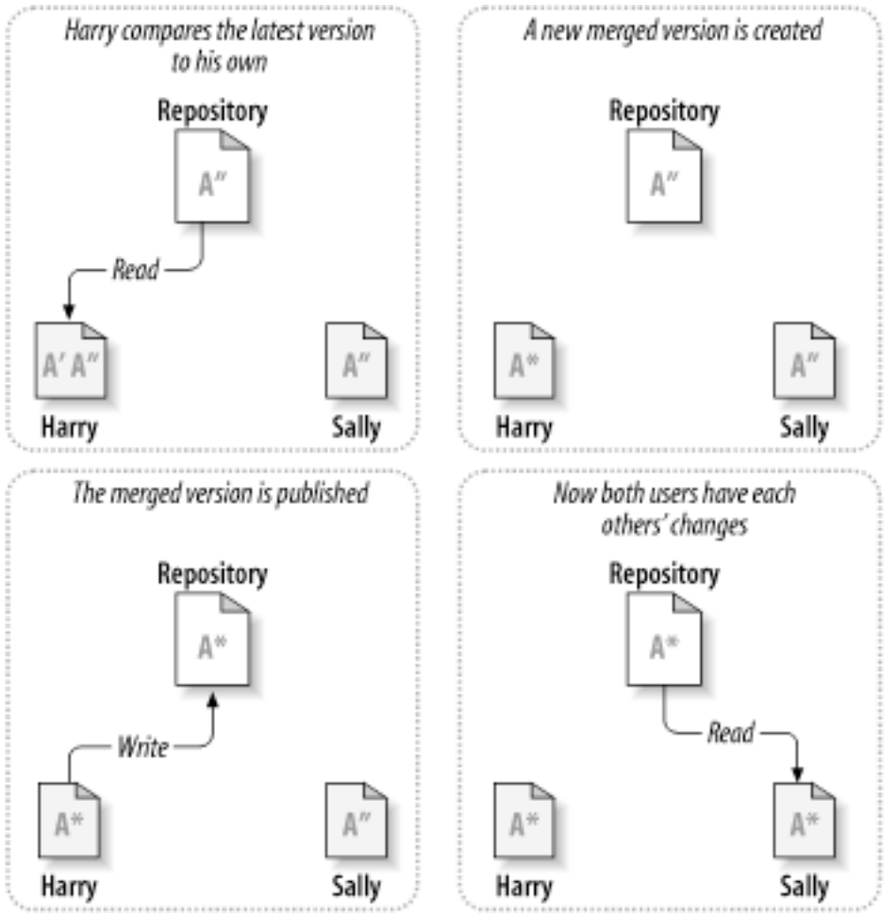
`svn:needs-lock`

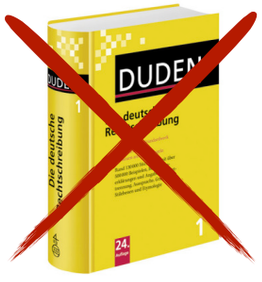
Konzept: gleichzeitige Änderungen : Auflösung

- Harry **update**t seine Arbeitskopie mit den neusten Änderungen vom Repository

- ... behebt evtl. **Konflikte**

- ... und **commit**et dann den zusammengeführten (ge**merge**ten) Stand

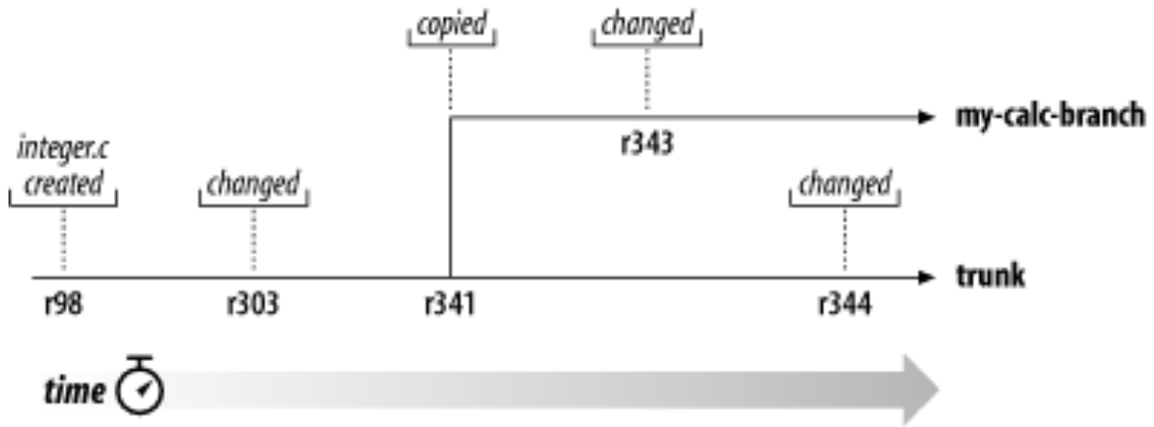


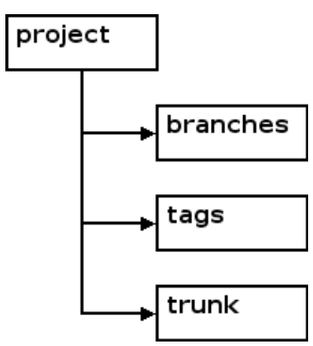


Konzept: Branching (Zweige)

- z. B. neue Features
- paralleler Entwicklungszweig mit gleicher Vorgeschichte
- Ein Branch ist nur ein Ordner

-> manuelles Zusammenführen der Änderungen





„**Merging changes** sounds simple enough, but in practice it can become a headache. The problem is that if you repeatedly merge changes from one branch to another, you might accidentally merge the same change *twice*. When this happens, sometimes things will work fine. When patching a file, Subversion typically notices if the file already has the change, and does nothing. But if the already-existing change has been modified in any way, you'll get a conflict.

Ideally, your version control system should prevent the double-application of changes to a branch. It should automatically remember which changes a branch has already received, and be able to list them for you. It should use this information to help automate merges as much as possible.

Unfortunately, Subversion is not such a system. Like CVS, Subversion does not yet record any information about merge operations. When you commit local modifications, the repository has no idea whether those changes came from running `svn merge`, or from just hand-editing the files.“

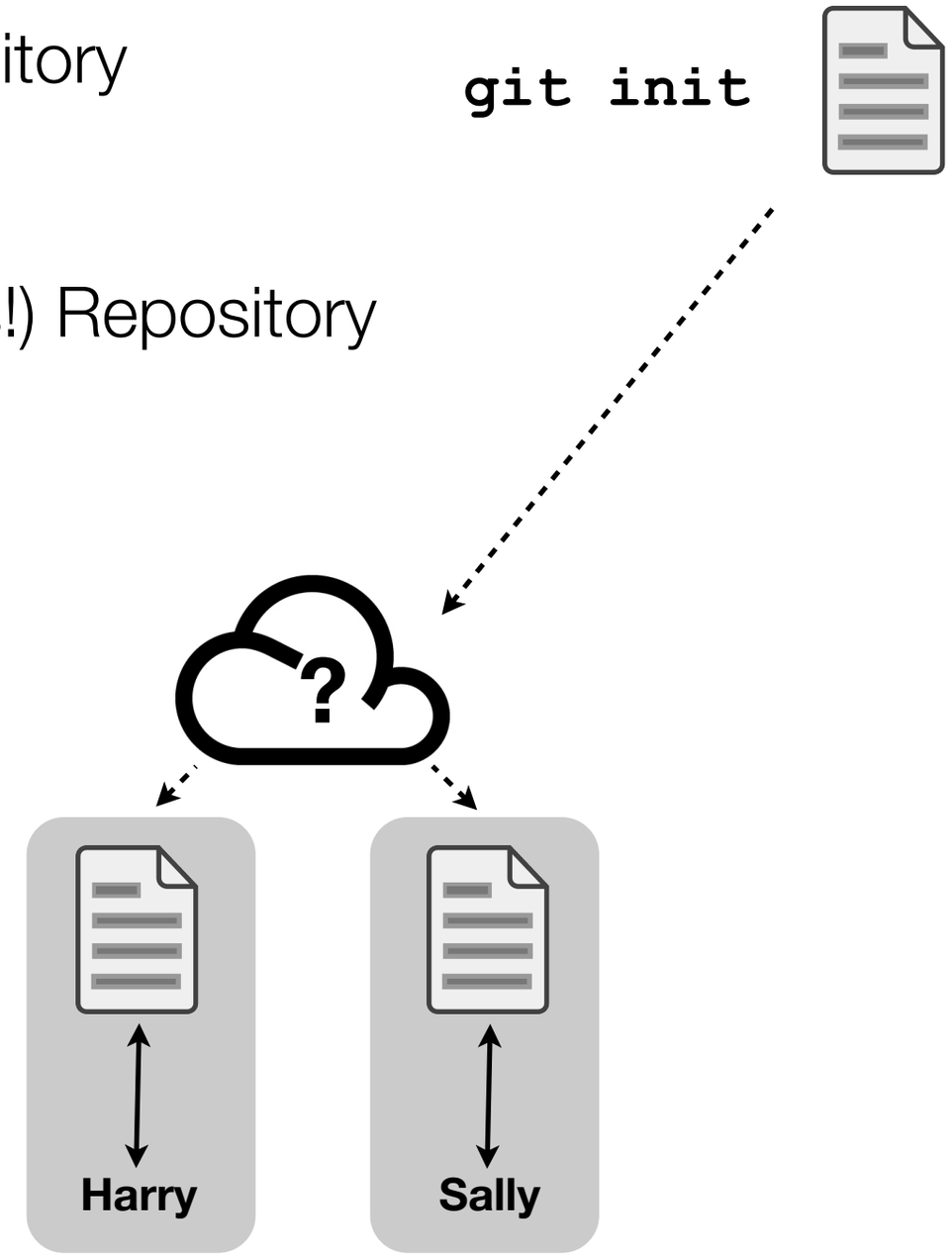
http://svnbook.red-bean.com/en/1.1/ch04s03.html

Konzept

- **dezentrales / verteiltes** = eigenes Repository
  - (Ordner, GIT-/SSH-/Web-Server)

- eigene Arbeitskopie = eigenes (vollwertiges!) Repository
  = eigener Entwicklungszweig (Branch)

- nach Fertigstellung werden die Änderungen **commit**et (lokal!)



- **Commit** bedeutet NICHT, dass es an einen Server gesendet wird

- Git hat eine Liste von entfernten Repositorys
  - push und pull

- „Ursprungsrepository“ (Vorlage des Klons) heißt **`origin`**

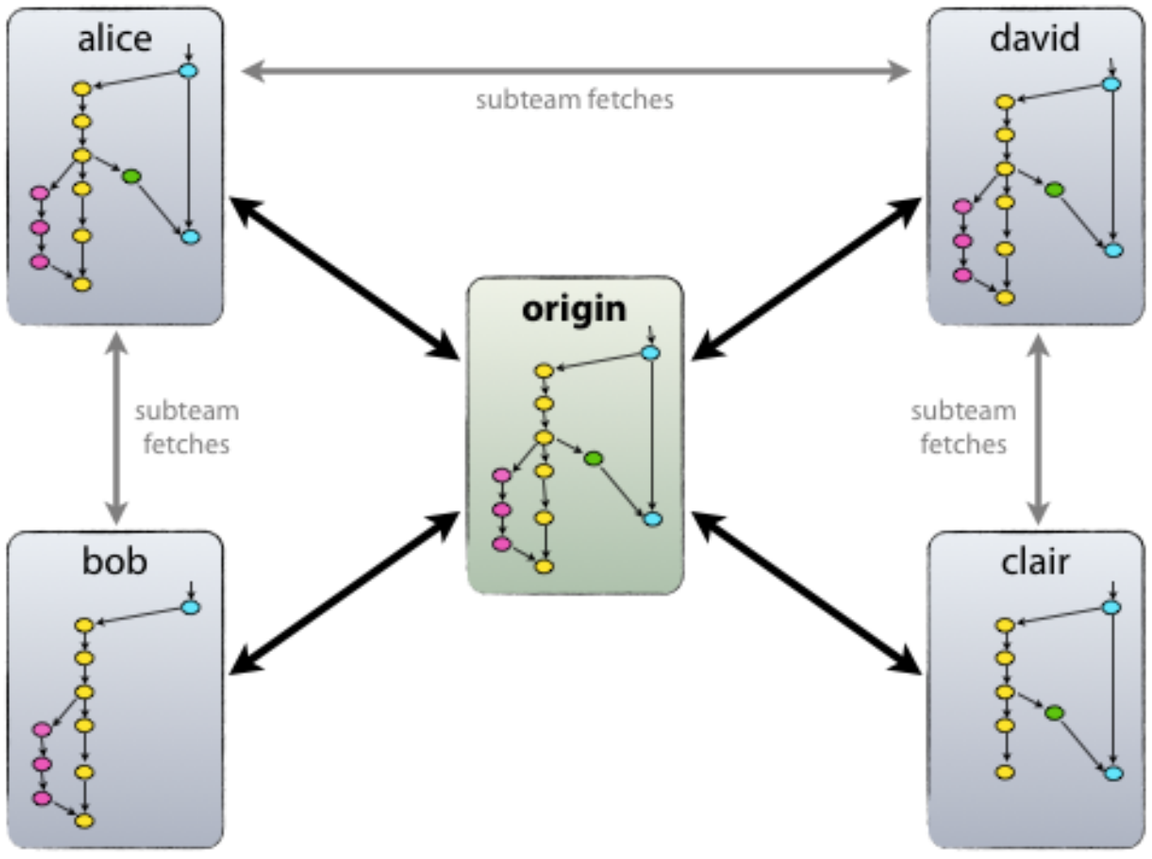
`git clone` *`url`*	`svn checkout` *`url`*
`git pull` (= fetch + merge)	`svn update`
`git commit -a && git push`	`svn commit`

Konzept: Pull/Push, Branching/Merging

- Benutzer arbeiten gleichzeitig an verschiedenen und gleichen Dateien

- Benuter „David“ arbeitet und commitet währenddessen mehrmals (lokal!)
  - nach Fertigstellung: **Push** an entferntes Repository (origin)

- Benutzer „Bob“ ist auch fertig, kann aber seine „lokalen Commits“ nicht an origin pushen.
  - er **pull**t die Änderungen von David und **push**t dann den (automatisch) ge**merge**ten Stand.

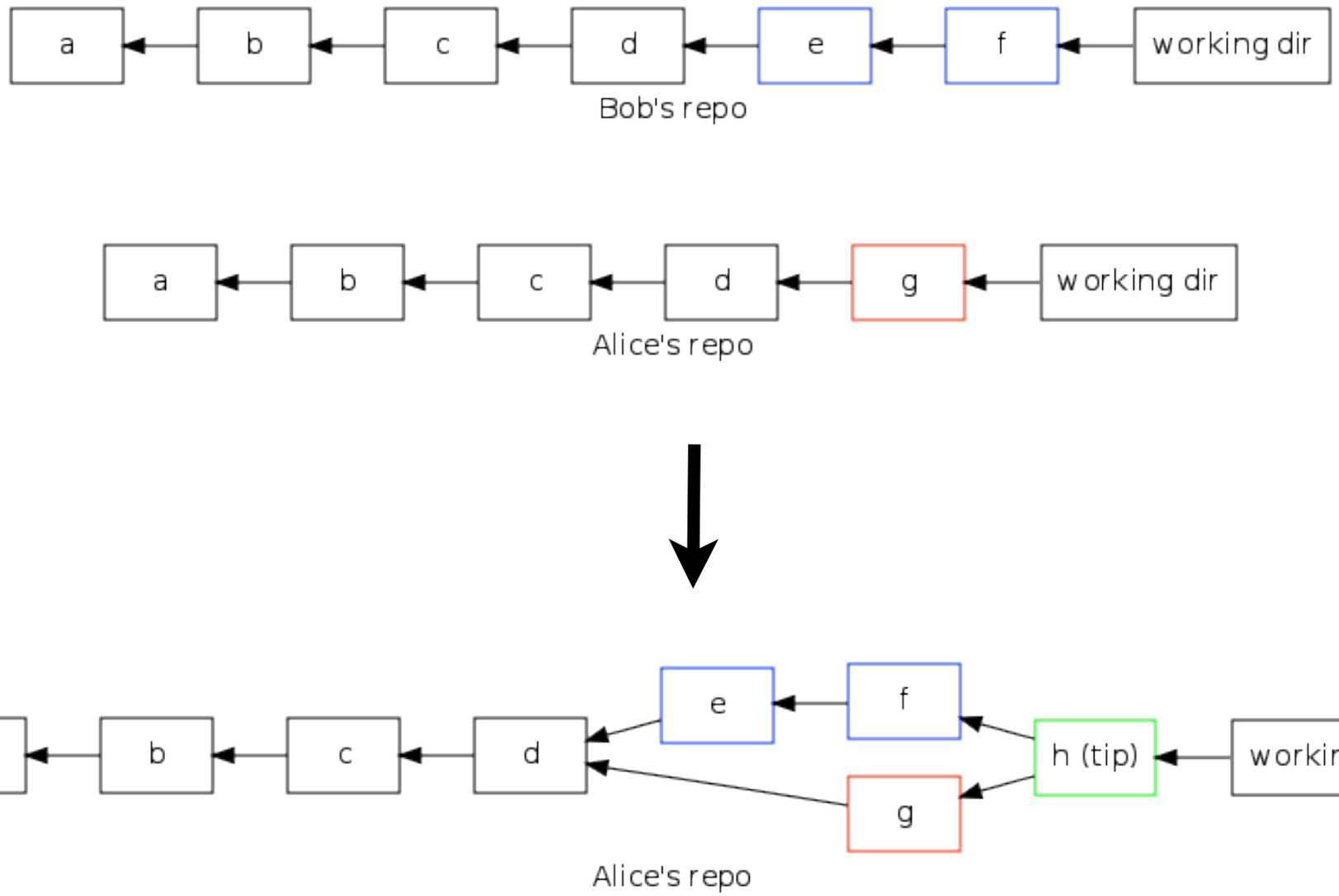


http://www.mageblog.de/wp-content/uploads/2012/06/centr-decentr.png

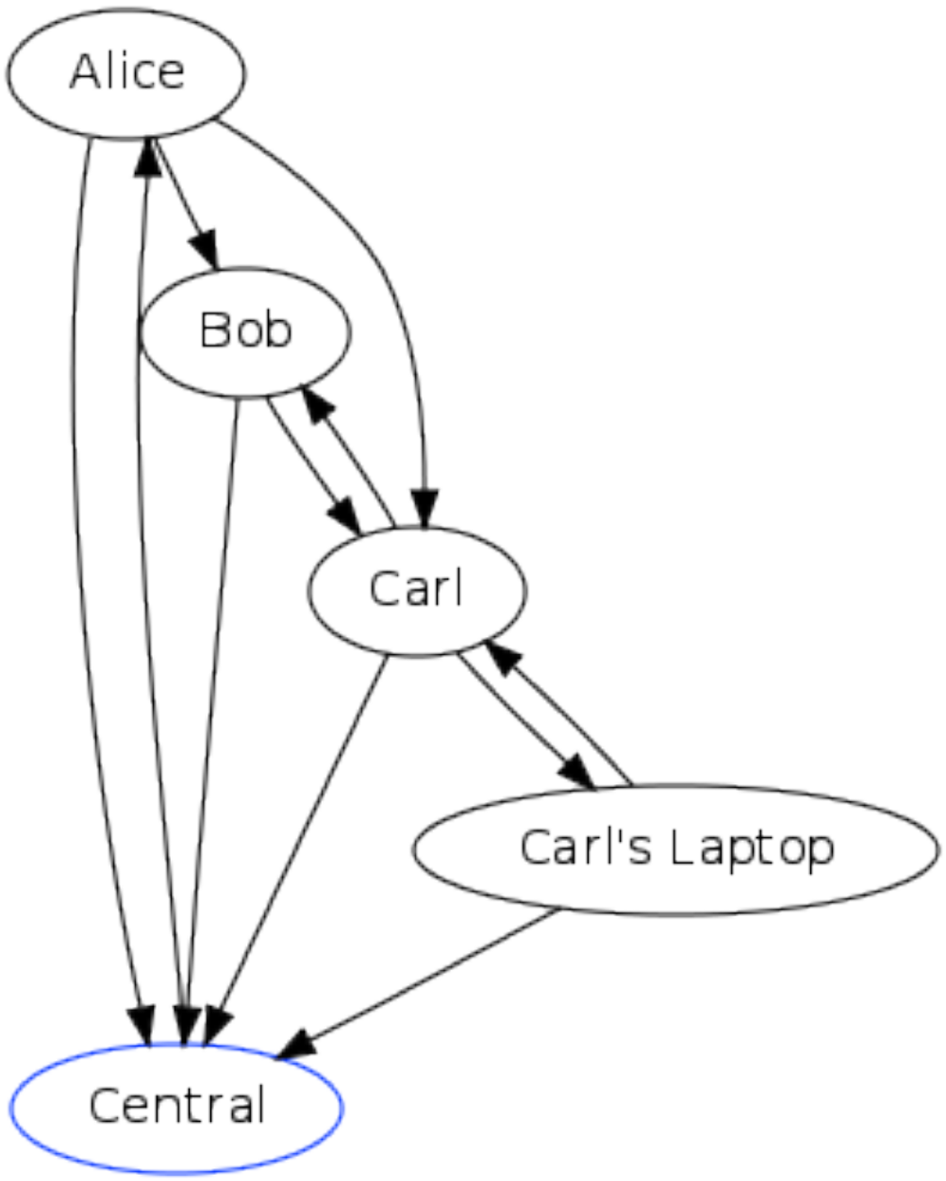
„Git's branching, tagging, merging, and rebasing are near flawless: git's merging algorithm is close to omniscient, having once merged 12 Linux kernel patches simultaneously“

http://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/

Konzept


a
b
c
d
e
f
working dir
Bob's repo
a
b
c
d
g
working dir
Alice's repo
a
b
c
d
e
f
g
h (tip)
working dir
Alice's repo

`git pull`	`hg fetch` `hg pull -u`
`git fetch`	`hg pull`
`git push`	`hg push -r .`

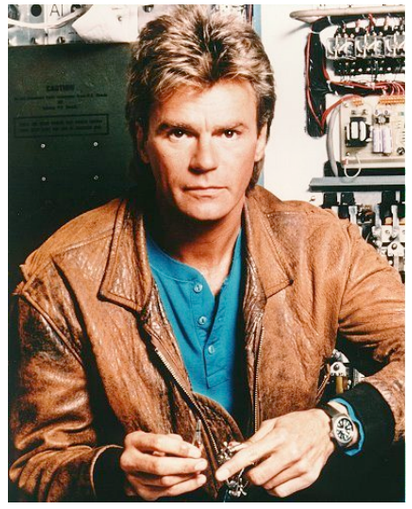


A Mercurial Network

http://mercurial.selenic.com/wiki/UnderstandingMercurial

Git vs. Hg — Mac Gyver vs. James Bond

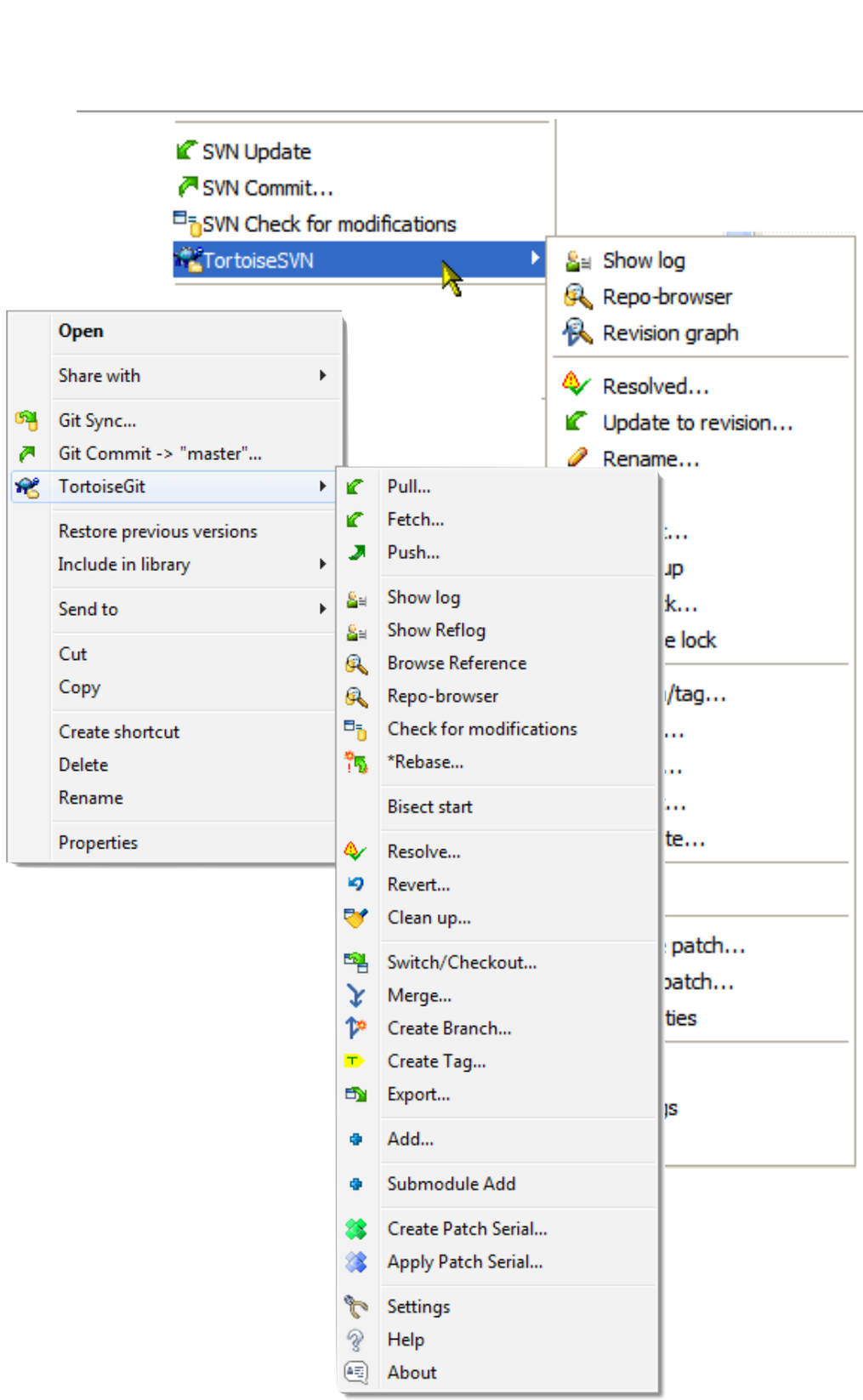
- **Git** ist eine Toolbox mit Werkzeugen zur Versionskontrolle
  - kommt mit jedem Workflow und in jeder Situation zurecht
  - (*wenn man den passenden Befehl kennt*)

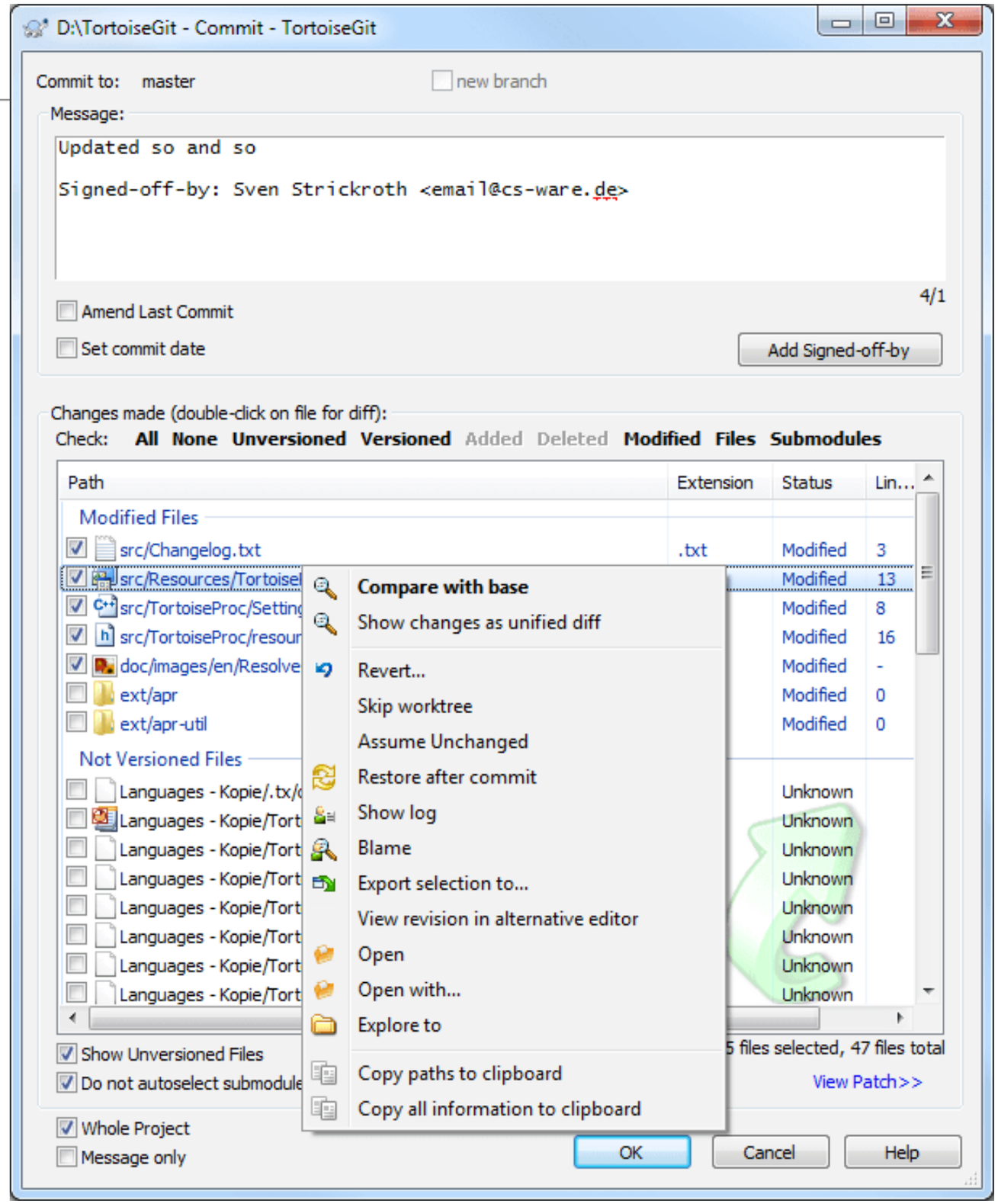


- **Mercurial** ist ein Werkzeug gemacht für einen Zweck
  - ähnlich wie Git, aber nicht so viele Freiheiten

GUIs und Setup

TortoiseSVN, TortoiseGit, TortoiseHG (Win)





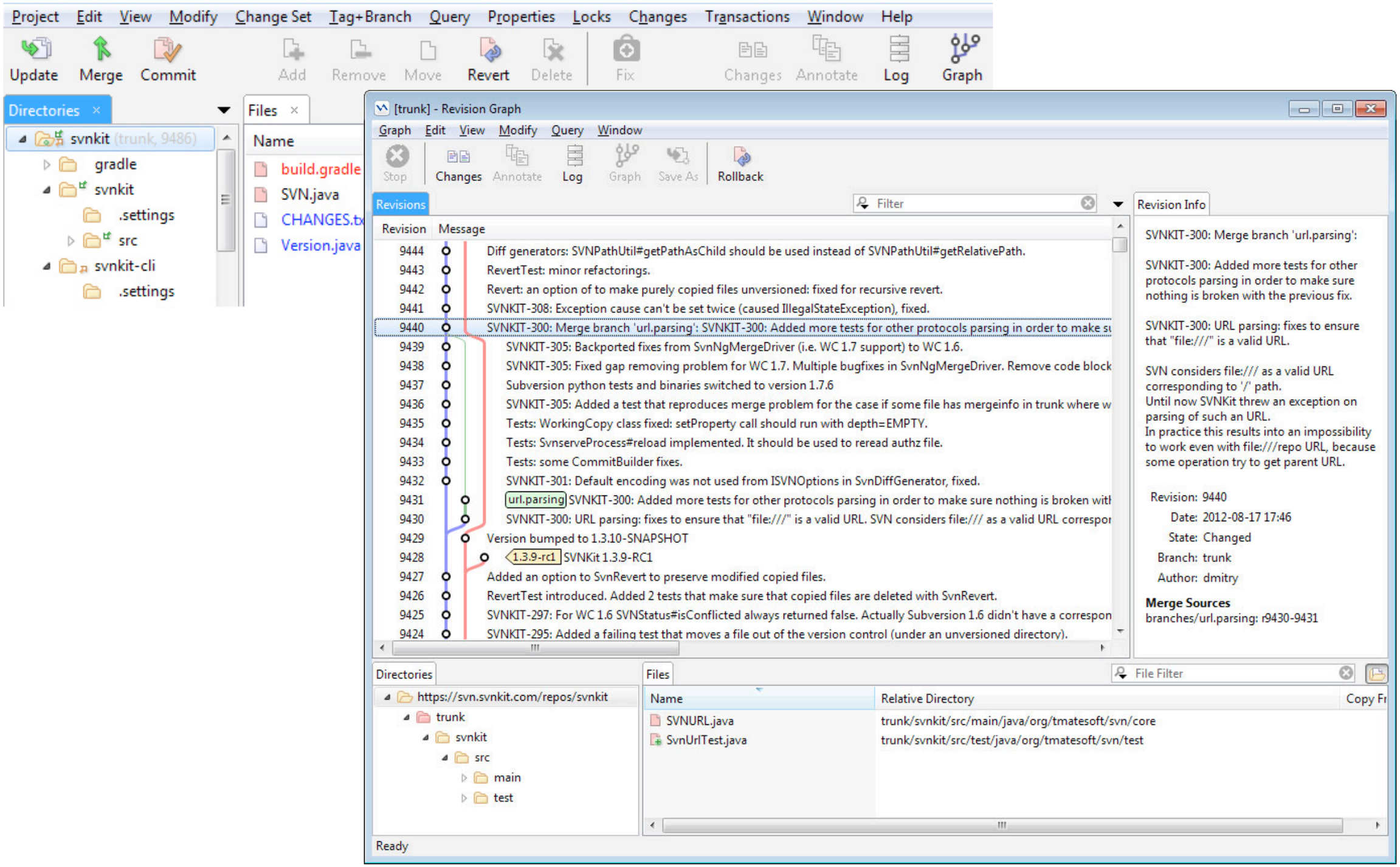
Subversion

- viele GUIs und Tools verfügbar

- vergleichsweise einfach aufzusetzen (CollabNet)

- zentraler Server zwingend notwendig

SmartSVN (Win, Mac, Linux)

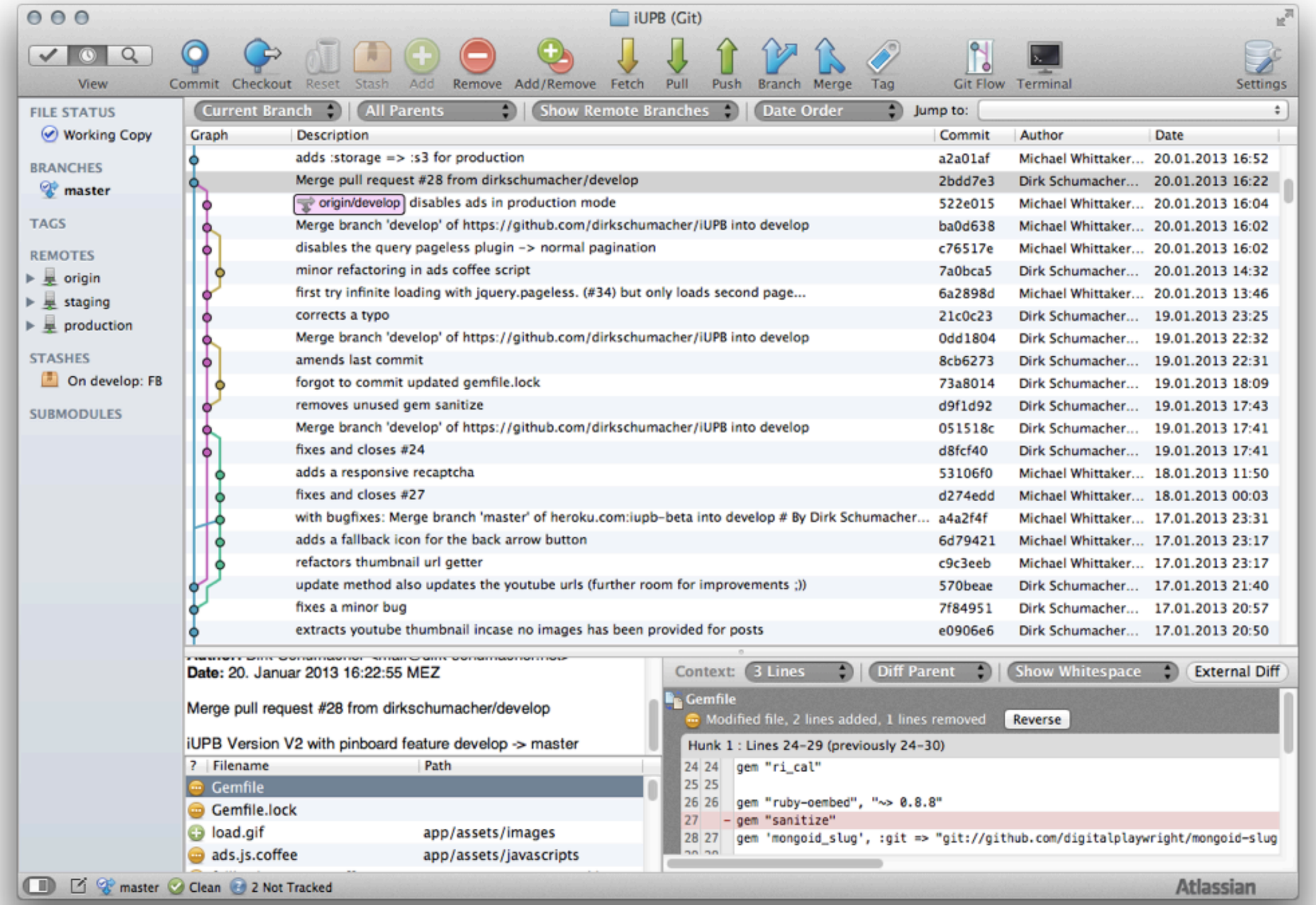


Git

- viele GUIs verfügbar, einige schöner als andere ;-)
- einfaches Setup
- zentrales Repository mit Gitolite, etc. einfach (unter UNIX) einzurichten
- Kommandozeile sehr mächtig

```
➜  ~  git int
WARNING: You called a Git command named 'int', which does not exist.
Continuing under the assumption that you meant 'init'
in 0.1 seconds automatically...
Initialized empty Git repository in /Volumes/Daten/Users/mwhittak/.git/
➜  ~ git:(master) ✗
```

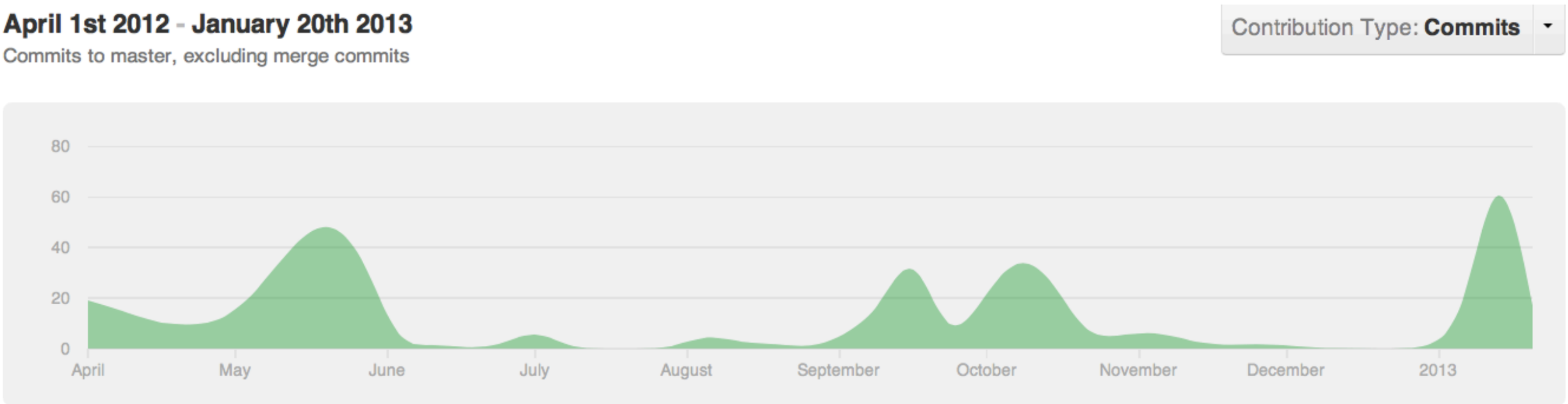
SourceTree (Mac)



git-bisect

http://software-as-a-craft.blogspot.de/2010/12/how-git-saved-my-bacon-today.html

mehr Commits = besser?



Da sind noch...

- CVS – nicht benutzen

- SVK – dezentrales SVN

- Bazaar - verteiltes VCS von Ubuntu-Hersteller

- weitere open-source und proprietäre Tools
  en.wikipedia.org/wiki/List_of_revision_control_software

Fazit

- Subversion sehr einfach zu nutzen, einfacher Workflow
  - viele GUIs, Tools und Integrationsmöglichkeiten
- Einfachheit von Subversion gleichzeitig größtes Argument *für* Git/HG
  - verteilt
  - „natürliches“ Braching und Merging

- **lineare und zentrale Entwicklung --> Subversion oder Git / Hg**
- **verteilte, sehr asynchrone oder „agile“ Entwicklung --> Git / Hg**

1. „Evaluate your workflow and decide which tool suits you best.
2. Learn how to use your chosen tool as well as you possibly can.
3. Help newbies to make the transition.
4. **Shut up about the tools you use and write some code.“**

http://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/

Das war‘s :-)

MichaelWhi online:

post@michael-whittaker.de

michael-whittaker.de

twitter.com/MichaelWhi

github.com/MichaelWhi

Folien von heute

Links

- Git Wiki: **Comparison with SVN**: git.wiki.kernel.org/index.php/GitSvnComparison
- **Sei (k)ein Blödmann und nimm Git**: slideshare.net/kogakure/sei-kein-bldmann-und-nimm-git-1830449
- **Hoster:**
  GitHub (www.github.com, 5 private Repos für Studenten: github.com/edu),
  BitBucket (www.bitbucket.org, private Repos gratis, Git und Hg),
  Uni Paderborn (IMT-Link 91411, SVN)
  Beanstalk (beanstalkapp.com, SVN, Git und Hg)
- **Selbst hosten**
  Git:
  Ordner+SSH, **Gitolite** (github.com/sitaramc/gitolite/wiki), **GitLab** (gitlabhq.com), **Gitorious** (gitorious.org/gitorious), **Stash** (atlassian.com/software/stash/)
  Subversion:
  SVN-Server, Apache-Setup, CollabNet (www.collab.net/products/subversion) , **Redmine** (redmine.org), …
- **GUIs**
  SourceTree (sourcetreeapp.com, Mac, Git+Hg), **SmartGit** (syntevo.com/smartgithg/, Git+Hg), **Tortoise** (gitorious.org/gitorious, Win, Git+Hg+SVN), **GitHub App** (mac.github.com / windows.github.com, Mac/Win), **Versions** (versionsapp.com, Mac, SVN, $), **SmartSVN** (smartsvn.com, alle, SVN), …
  EGit for Eclipse, **MercurialEclipse**, **GitHub for Eclipse**, …

Links II

- **SVK:** svk.bestpractical.com
- **Bazaar:** bazaar.canonical.com

- **iUPB**: www.i-upb.de (Code [und Commits]: github.com/yippie-io/iUPB)

BONUS: Commit Messages – „Fool a tool“

Pflicht bei Git (<=> SVN), aber... :)

May 09, 2012

Update App/Github_scanner_login/src/linkdroid/login/test/Remote_help.... ...
lindentwig authored 9 months ago — 6bcf01688a — Browse code

Apr 25, 2012

Update README
lindentwig authored 9 months ago — dafd2ea914 — Browse code

complete session
Andreas Lindqvist authored 9 months ago — 21c8f99147 — Browse code

Apr 21, 2012

Update README
lindentwig authored 9 months ago — 9844bf4d8f — Browse code

Apr 18, 2012

Update Project_log
lindentwig authored 9 months ago — d681c6aba3 — Browse code

Update README
lindentwig authored 9 months ago — 0f48053dd1 — Browse code

Apr 12, 2012

Update README
lindentwig authored 10 months ago — 26831b359f — Browse code

first real commit
Andreas Lindqvist authored 10 months ago — a5302e3b5e — Browse code

Apr 10, 2012

Merge branch 'master' of github.com:lindentwig/Android-QR-code-challe... ...
Andreas Lindqvist authored 10 months ago — c49d951eb8 — Browse code

Apr 03, 2012

first code commit
Andreas Lindqvist authored 10 months ago — b4b70e3a90 — Browse code

bessere Idee: github.com/erlang/otp/wiki/Writing-good-commit-messages