



Konferenz für Java™, Enterprise Architekturen, SOA

Dr. R. Guderlei | eXXcellent solutions gmbh  
Tim Felgentreff | HPI

# Versionsmanagement

Zentral oder Verteilt?

# Agenda

- Verteilte Versionsverwaltung mit Git
- Git in der Praxis
- Fazit

# Grundlegendes

- Verteilung: kein zentrales Repository
- Jeder Entwickler hält lokal eine Kopie des Repositories

# Vorteile

- Offline arbeiten möglich
- Daten werden redundant gehalten (Datenverlust weniger wahrscheinlich)
- Hohe Geschwindigkeit der Operationen
- Häufigere, kleinere Commits
- Es wird mehr mit (lokalen) Branches gearbeitet

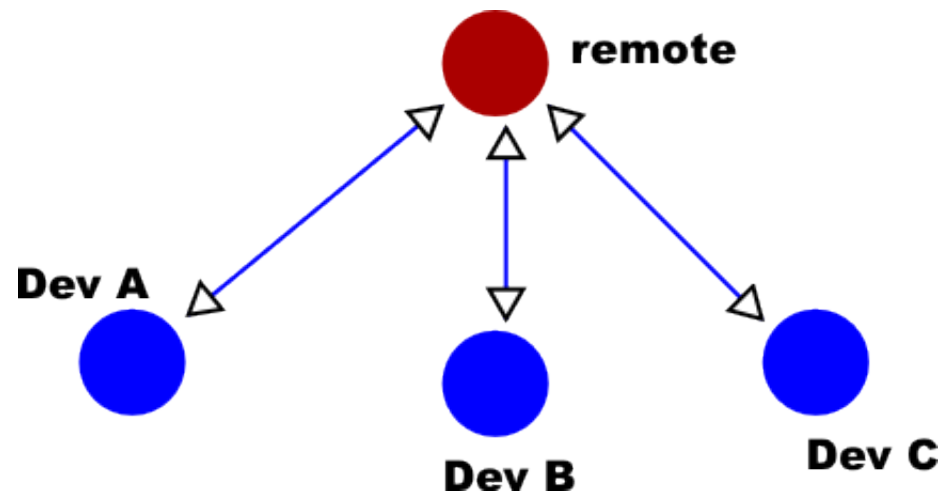
# Beispiele

- Kommerziell
  - BitKeeper
  - ClearCase
- Open Source
  - Darcs
  - Bazaar
  - Mercurial
  - **Git**
  - ...

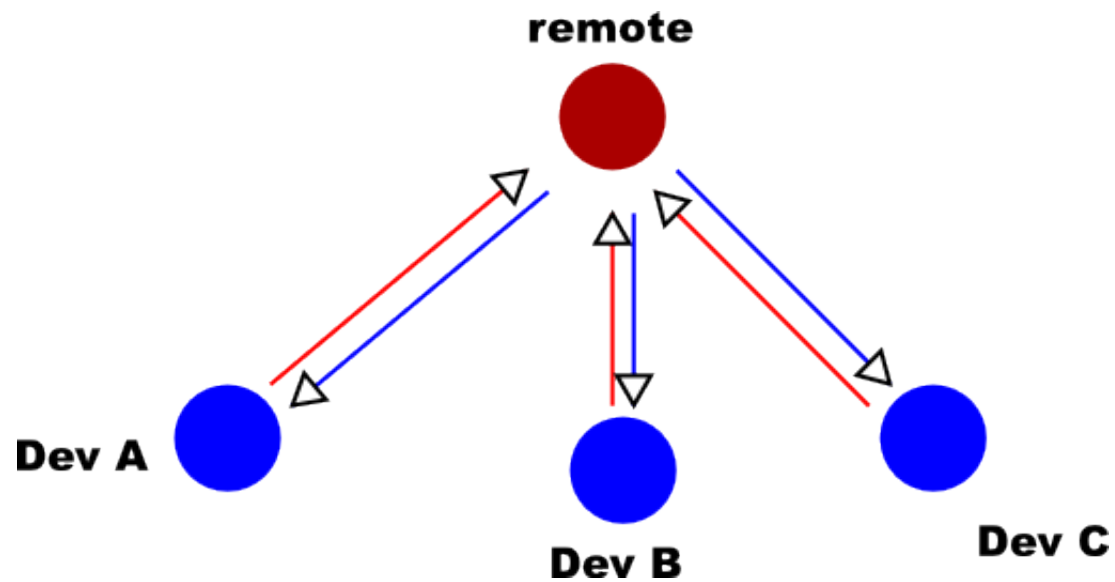
# Warum Git und nicht xyz?

- Erprobt in großen Projekten (Linux-Kernel, QT, Ruby on Rails, Gnome, Android, X.org, ...)
- Geschwindigkeit
- Tool-Support (IDEs, TortoiseGit...)
- GitHub

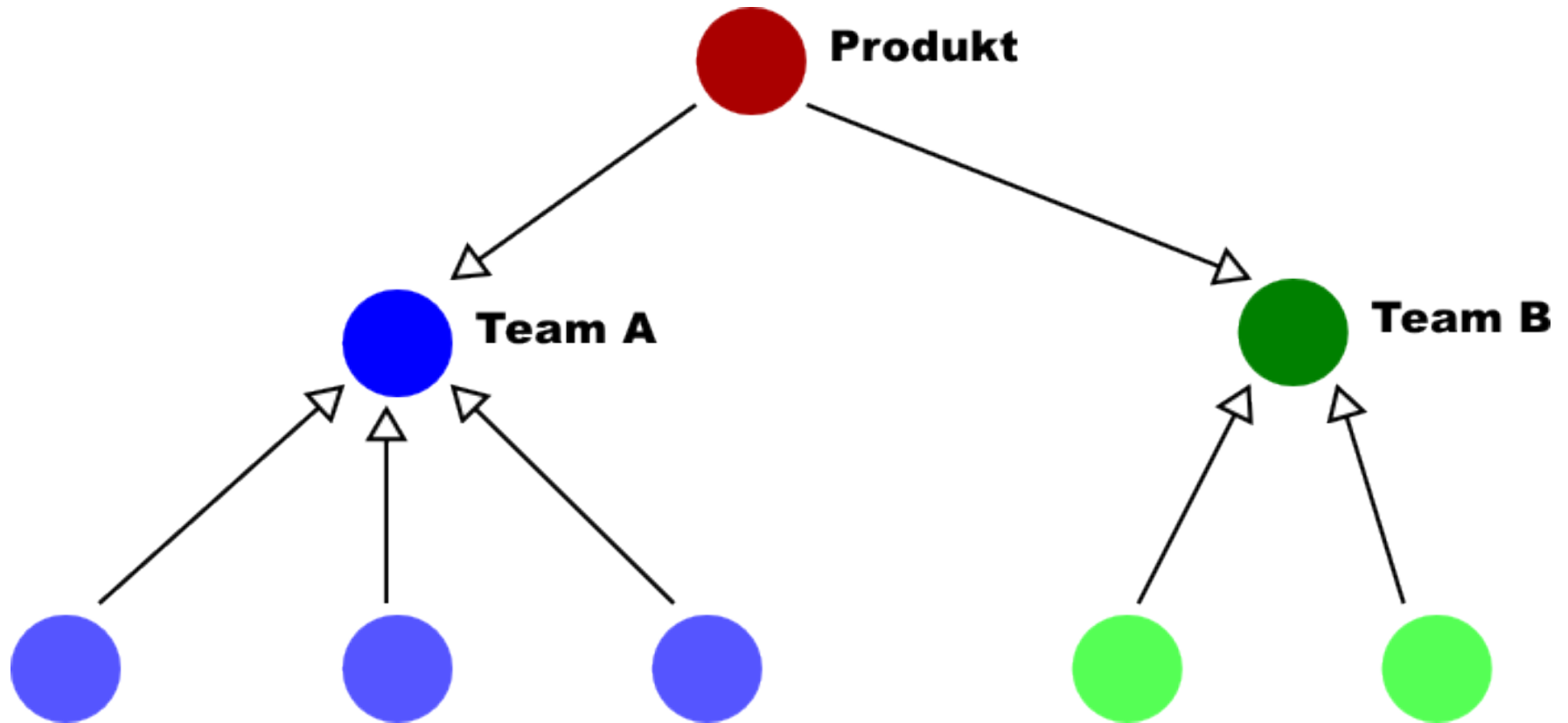
# „SVN-ähnliche“ Struktur



# „pull only“-Struktur



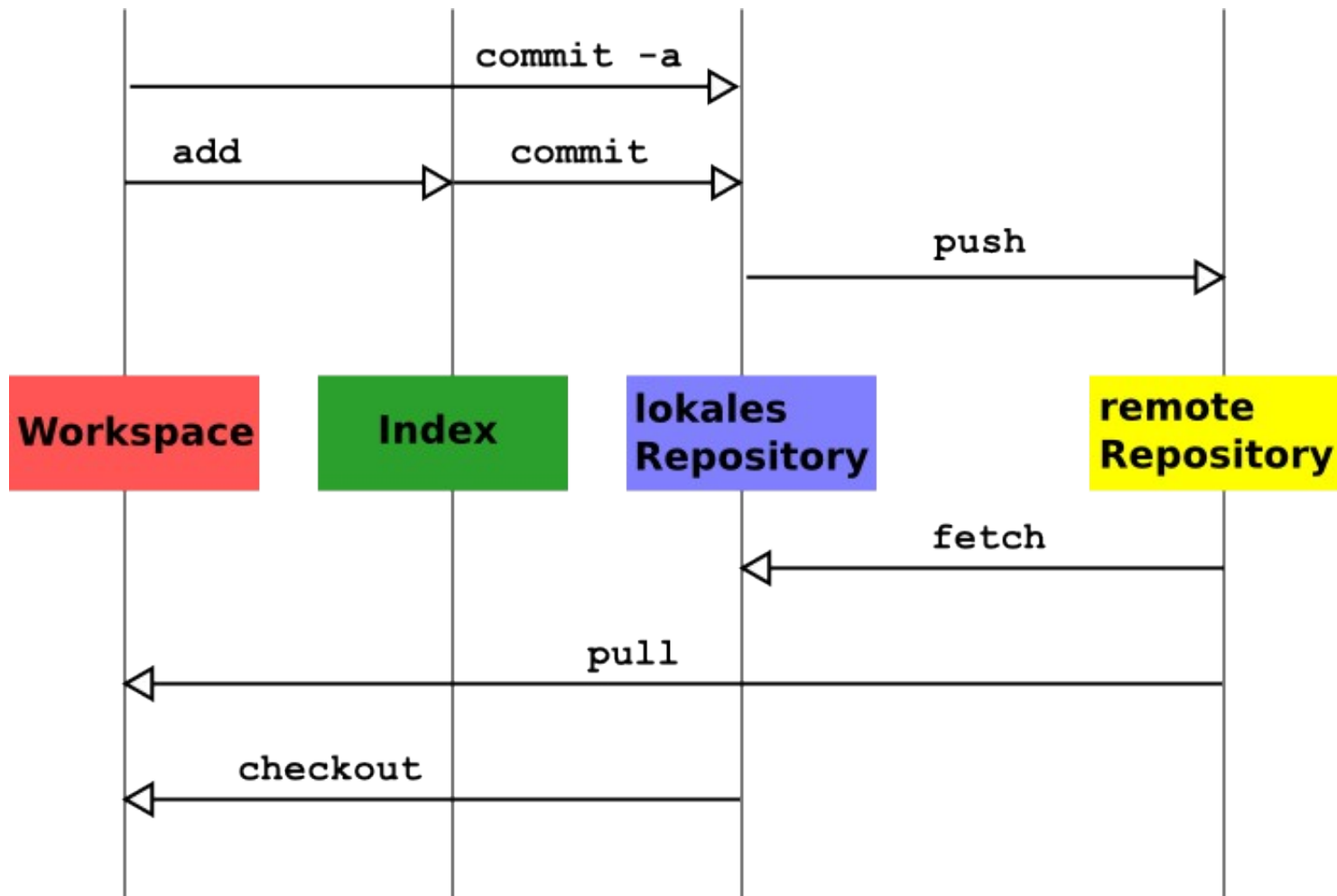
# Repo-Hierarchien



# Betrieb

- Default: Zugriffssteuerung über Systemberechtigungen (git+ssh)
- Rechtemanagement: Gitis  
(<http://eagain.net/gitweb/?p=gitoris.git>)
- Reviews: Gerrit  
(<http://code.google.com/p/gerrit/>)
  - Eigene Rechteverwaltung
  - Code - Reviews
- Privates Git Hosting: Gitorious, Bithug  
(<http://gitorious.org/>,  
<http://github.com/rkh/bithug>)

# Befehle



# Commit Messages

[PRJ-4711] Tests für XYZ hinzugefügt

Folgende Tests sind neu dazugekommen:

- \* Szenario A
- \* Szenario B
- \* Prüfung auf ungültigen Parameter

# „SVN-Mode“

```
rguderlei@n149: ~/projects/osgi/osgi-demo
Datei Bearbeiten Ansicht Terminal Hilfe
rguderlei@n149:~/projects/osgi/osgi-demo$ git pull
Already up-to-date.
rguderlei@n149:~/projects/osgi/osgi-demo$ vim README
rguderlei@n149:~/projects/osgi/osgi-demo$ git commit -am "Maven version in README"
[master 46d84bb] Maven version in README
 4 files changed, 7 insertions(+), 9 deletions(-)
rguderlei@n149:~/projects/osgi/osgi-demo$ git push
Counting objects: 27, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (14/14), 1.00 KiB, done.
Total 14 (delta 6), reused 0 (delta 0)
To git@github.com:rguderlei/osgi-demo.git
 ecf3475..46d84bb  master -> master
rguderlei@n149:~/projects/osgi/osgi-demo$
```

# Branch/Rebase

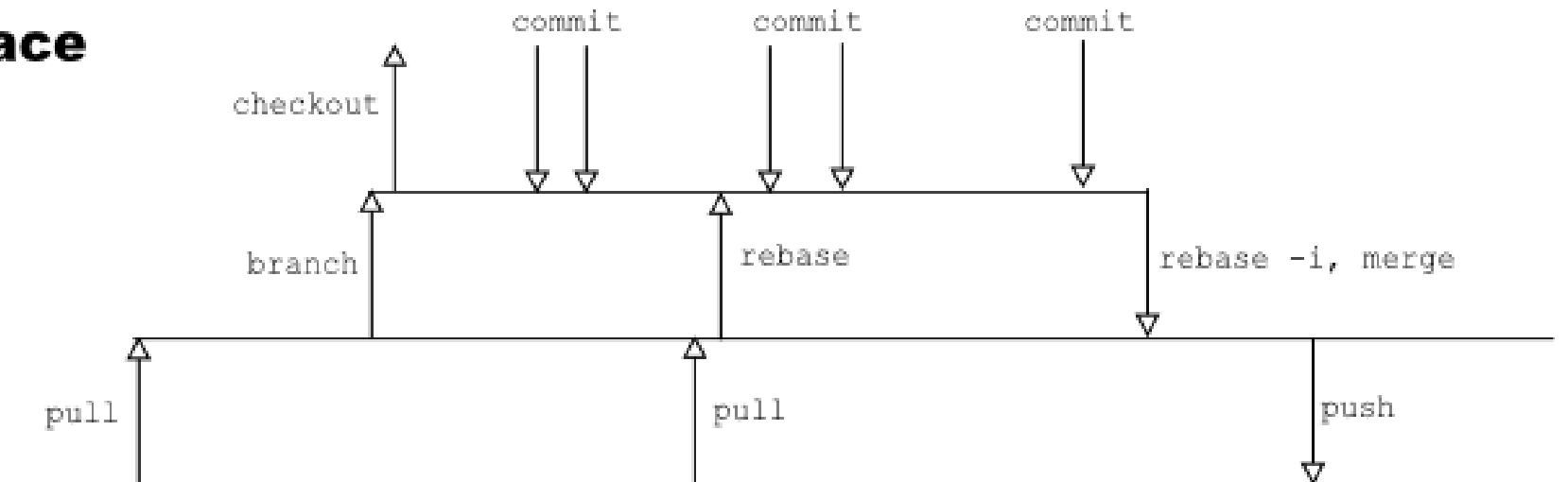
1. Neuer Branch für Feature/Bug
2. Arbeit im Branch
3. Rebase: Änderungen am Master-Branch regelmäßig integrieren
4. Branch in Master mergen, ggf. Commits zusammenfassen
5. push

## workspace

## branch

## master

## origin



# Squashing

- Mit `rebase --interactive` kann man
  - Mehrere Commits zusammenfassen (squash)
  - Ein Commit in mehrere Commits zerlegen
  - Die Reihenfolge von Commits verändern
- **NIEMALS** mit Commits machen , die sich bereits in einem remote repository befinden. **NEVER EVER.** (es sei denn man zwingt jeden Entwickler zu einem ‚git pull -force‘)

# Git - rename

- Git unterstützt keine „intelligent renames“
- Heuristische Erkennung von renames  
→ renames sofort committen
- Bewusste Designentscheidung  
→ Tradeoff mit Geschwindigkeit

# Sonstige Features

- Stashing: temporäres Lagern von Änderungen
- Patches per Mail
- Bisect : Suche nach Commits (z.B. beim Bugs suchen)
- Cherry Picking: selektiv Commits übernehmen

*And then realize that nothing is perfect. Git is just  
\*closer\* to perfect than any other SCM out there.*

- Linus Torvalds



# Tool Support

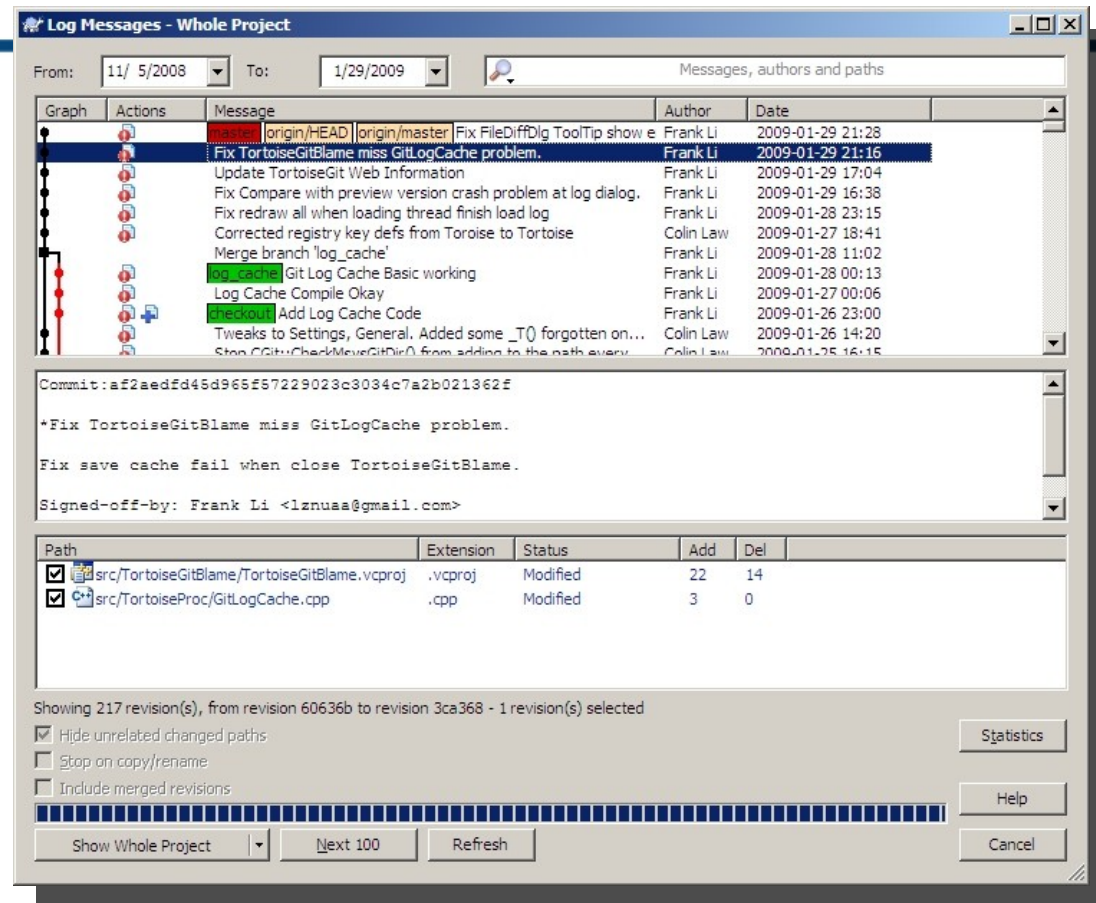
- TortoiseGit für Windows

- TortoiseSVN Port für msys-git

- Gut geeignet für Umsteiger

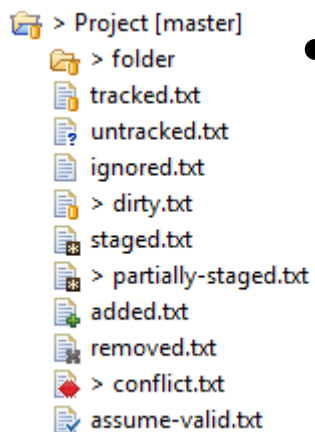
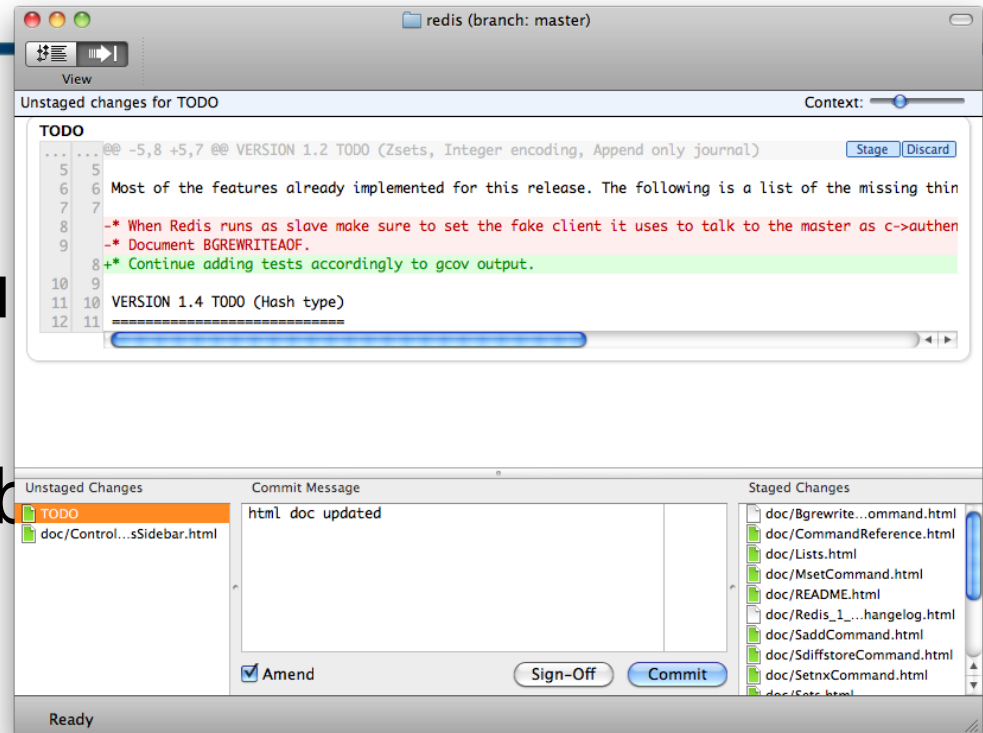
- Gute Integration mit dem Explorer

- Kein Support für komplexes Staging mit Line Commits



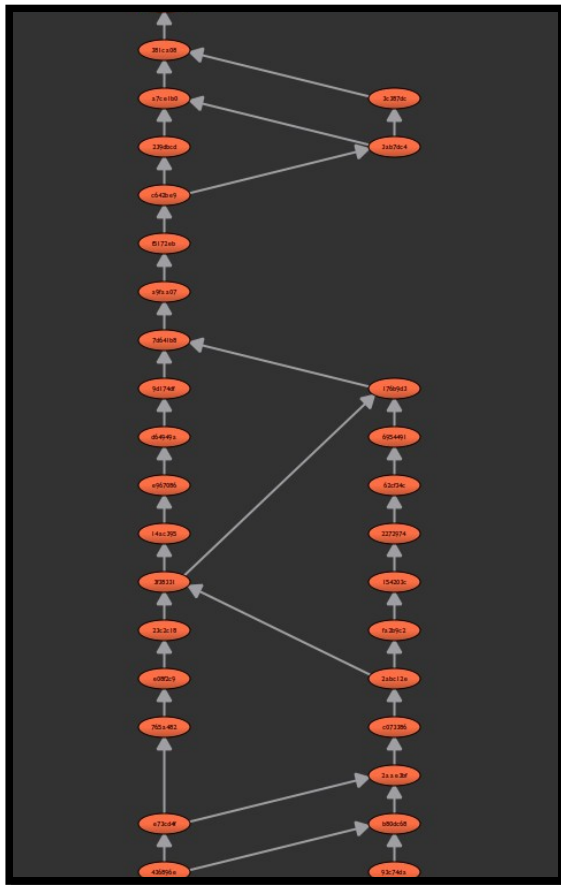
# Tool Support

- GitX für Mac OS X
  - Sehr guter Support für komplexes Staging
  - QuickLook und GitHub Integration



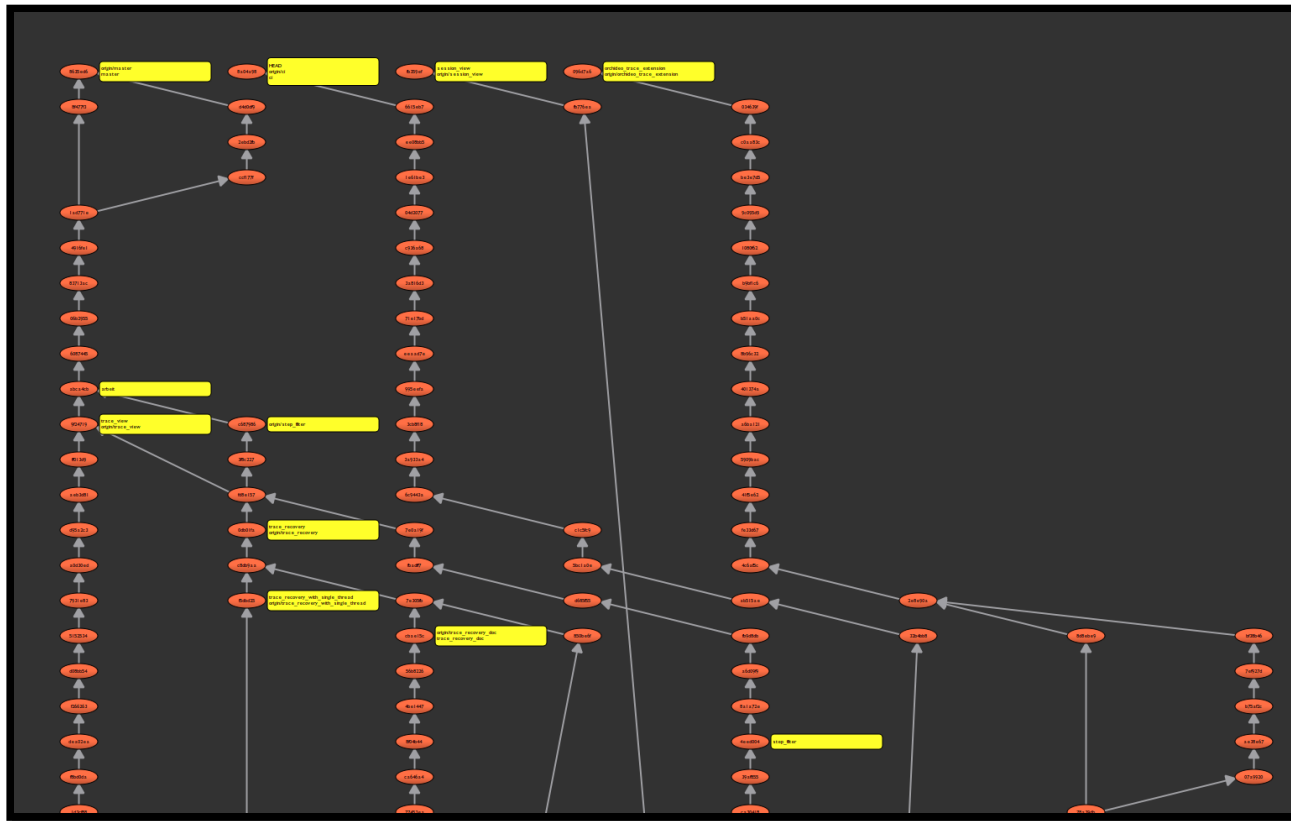
- EGit für Eclipse
  - Eclipse integration wie Subversive
  - Amend Commit Support
  - Kein Support für Line Commits

# Der svn-Workflow



- ▶ Linearer Versionsverlauf
- ▶ Lange SVN Historie
- ▶ Lineares Branching für Features
- ▶ Kaum Parallele Entwicklung auf mehreren Branches

# Der git-Workflow



- ▶ Feature, Feature-stable und Feature-bug-#XYZ Branches
- ▶ Parallele Entwicklung nach Motivation und Zeitdruck
- ▶ Stets stabile Branches
- ▶ Cherry-Picking einzelner Commits
- ▶ Merges von Tags aus Feature Branches
- ▶ Merges von Bugfixes in Stable Branches

# Migration: SVN → git

```
File Edit View Terminal Help
$ git svn clone svn+ssh://svn.dev.company.org/
$ git remote add origin git://git.dev.company.org/
$ git push origin master
```

# git-svn

- Der Einstieg in den Umstieg
  - `svn co => git svn clone`
  - `svn up => git svn rebase`
  - `svn ci => git svn dcommit`
  - `svn cp => git svn branch`
  - Lokale commits, rebases und branches
- Gotchas
  - Commit Reihenfolge muss Upstream erhalten bleiben => Niemals mergen
  - Rebase weniger mächtig als Merge

# Erfahrungen im Umstieg

stash

diff

push

bisect

pull

cherry-pick

merge

log --graph



# Fazit

- **Kaum Nachteile**
  - Komplexere Prozesse
  - Relativ schlechte Windows-Integration
- **Viele Vorteile** gegenüber zentralen SCM
  - Mehr Flexibilität im Workflow
  - Robusteres System
  - „besseres“ Arbeiten
  - Integration bestehender SCMs (SVN, hg)
  - GitHub (<http://www.github.com>)
  - Wahl des DVCS hängt von Features ab (siehe intelligent renames)

Vielen Dank!  
Fragen?

Gerne auch am Stand von  
eXXcellent solutions  
im Ausstellungsbereich

# Erfahrungen im Umstieg

- Probleme liegen zu Beginn im scheinbaren Unwissen von „was ist aktuell?“ und dem „SVN-Workflow“
- Schritte zur Lösung:
  - Exzessives Branching und Merging
  - Tagging an Abstimmungspunkten
  - Engere Zusammenarbeit mit Quer-Merging zwischen Team-Mitgliedern um aktuell zu bleiben

# Git

- Entwickelt von Linus Torwalds
- Ersatz für BitKeeper für die Kernel-Entwicklung
- Einsatz seit 2005

# Features / Ziele

- Verteilte Entwicklung ermöglichen
- Hohe Geschwindigkeit
- Verwaltung großer Projekte
- Kryptographische Sicherung der Änderungshistorie
- Verwendung von Standard-Protokollen (ssh, http(s))