

Computergrundlagen Versionskontrollsysteme

Institut für Computerphysik
Universität Stuttgart

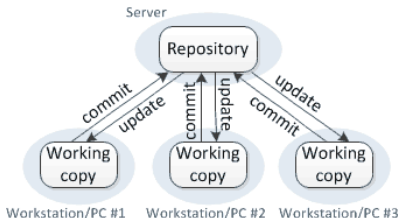
Wintersemester 2015/16

Versionsverwaltung

- zur Erfassung von Änderungen an Dokumenten oder Dateien
- Speichern platzsparend ältere Versionen von Dateien
- Speichern Informationen zu den Versionen
- Erlauben verteilten Zugriff
- Erleichtern gleichzeitiges Arbeiten an mehreren Versionen
- Versuchen, gleichzeitige Änderungen an Dateien zusammenzuführen
- Beispiel: Wikipedia
- Server-basiert oder verteilt

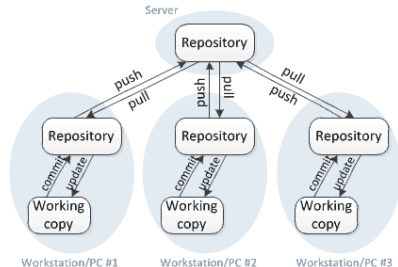
Versionskontrollsysteme (VCS)

Centralized version control



Zentrale Versionskontrolle
(z.B. SVN, CVS ...)

Distributed version control



Verteilte Versionskontrolle (z.B. Git, Mercurial ...)

Quelle: homes.cs.washington.edu

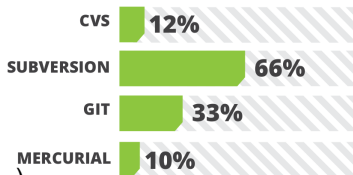
Anwendungsgebiete

- Softwareentwicklung
- Gemeinsame Arbeit an Dokumenten
- Regelmäßige Sicherung eigener Arbeiten

Hauptaufgaben

- Protokollierungen der Änderungen
- Wiederherstellung von alten Ständen einzelner Dateien
- Archivierung der einzelnen Stände eines Projektes
- Koordinierung des gemeinsamen Zugriffs von mehreren Entwicklern auf die Dateien
- Gleichzeitige Entwicklung mehrerer Entwicklungszweige (branches) eines Projektes.


Versionskontrollsysteme



Quelle: zeroturnaround.com (2013)

- CVS (<http://www.nongnu.org/cvs>)
 - Server-basiert (auch lokal über Dateien)
 - Älter, aber einfach zu bedienen
- Subversion (<http://subversion.tigris.org>)
 - Server-basiert (auch lokal über Dateien)
 - Ähnlich CVS, unterstützt mehr Funktionen
- GIT (<http://git-scm.com>)
 - Verteiltes Versionskontrollsystem
 - Erlaubt auch lokal in einem Verzeichnis mehrere Entwicklungszweige
 - VCS der Linux-Kernel-Entwicklung
- Mercurial (<http://mercurial.selenic.com>)
 - Verteiltes Versionskontrollsystem wie GIT
 - Wird z.B. auf Google Code angeboten (neben Subversion)

Grundlegendes Vorgehen

- Anlegen eines **Repositories** (Datenbank mit Historie)
 - Eventuell Anlegen einer Kopie des Verzeichnisses (**Checkout** einer **Arbeitskopie**)
 - Ändern von Dateien im Verzeichnis
 - Holen von Änderungen aus anderen Quellen (**Update**)
 - Diese werden mit den eigenen Änderungen verschmolzen (**Merge**)
 - Dabei kann es zu nicht automatisch auflösbaren Überschneidungen kommen (**Conflict**)
 - Diese müssen manuell behoben werden
 - Übergabe der Änderungen an das Repository (**Commit**)
 - Eventuell Erstellen von Entwicklungszweigen (**Branches**)
- 

Beispiel: GIT lokal

- Anlegen eines neuen Repositories: **git init Repo-Name**
 - Dies legt ein Verzeichnis *Repo-Name* an
 - Darin können Dateien und Verzeichnis wie gewohnt angelegt werden (Ausnahme: das `.git`-Verzeichnis nicht anfassen!)
- **git status**: Status der Dateien im Verzeichnis
 - Dateien sind erst einmal nicht verfolgt (**untracked**), also nicht versioniert
 - Liste aller geänderten versionierten Dateien
- Hinzufügen zur Versionskontrolle durch **git add Datei ...**
- **git commit -a** speichert alle Änderungen als neue Version
 - Erwartet einen Kommentar, damit man später weiss, was man committet hat
 - Man kann auch einzelne Dateien committen
- **git log** listet alle Commits und vor allem ihre IDs Nur der Anfang von *ID* ist nötig, solange das eindeutig ist