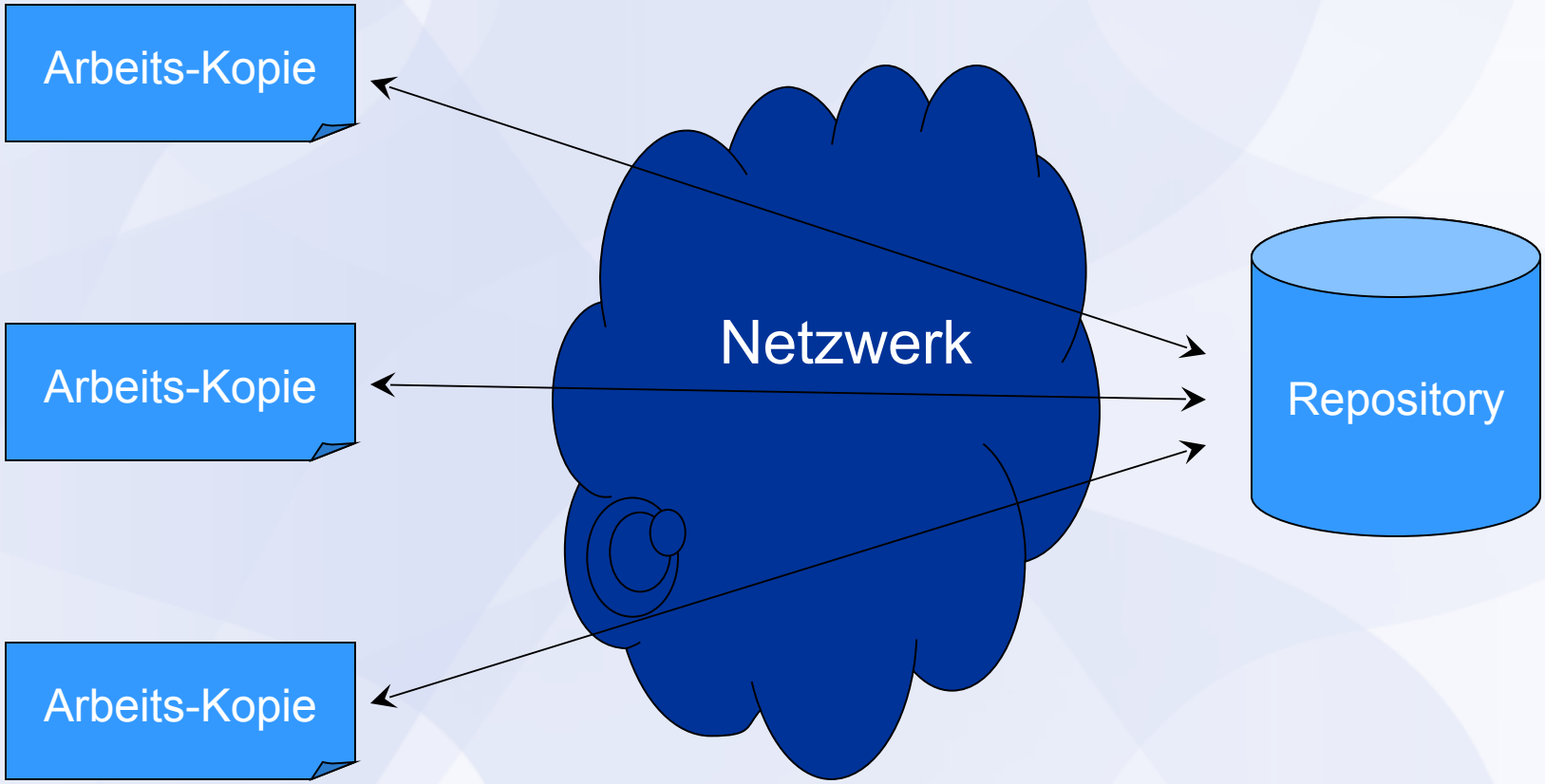
Einführung in Subversion

Benjamin Seppke

AB KOGS
Dept. Informatik
Universität Hamburg

Was ist Subversion?

- Ein Server-basiertes Versions-Verwaltungs-System
- Ermöglicht mehreren Benutzern die gemeinsame Arbeit an Verzeichnissen, Dateien und zeichnet dabei automatisch Änderungen auf.
- Subversion kann generell mit jeder Art von Datei umgehen (nicht nur Quelltext).


Arbeits-Kopie
Arbeits-Kopie
Arbeits-Kopie
Netzwerk
Repository

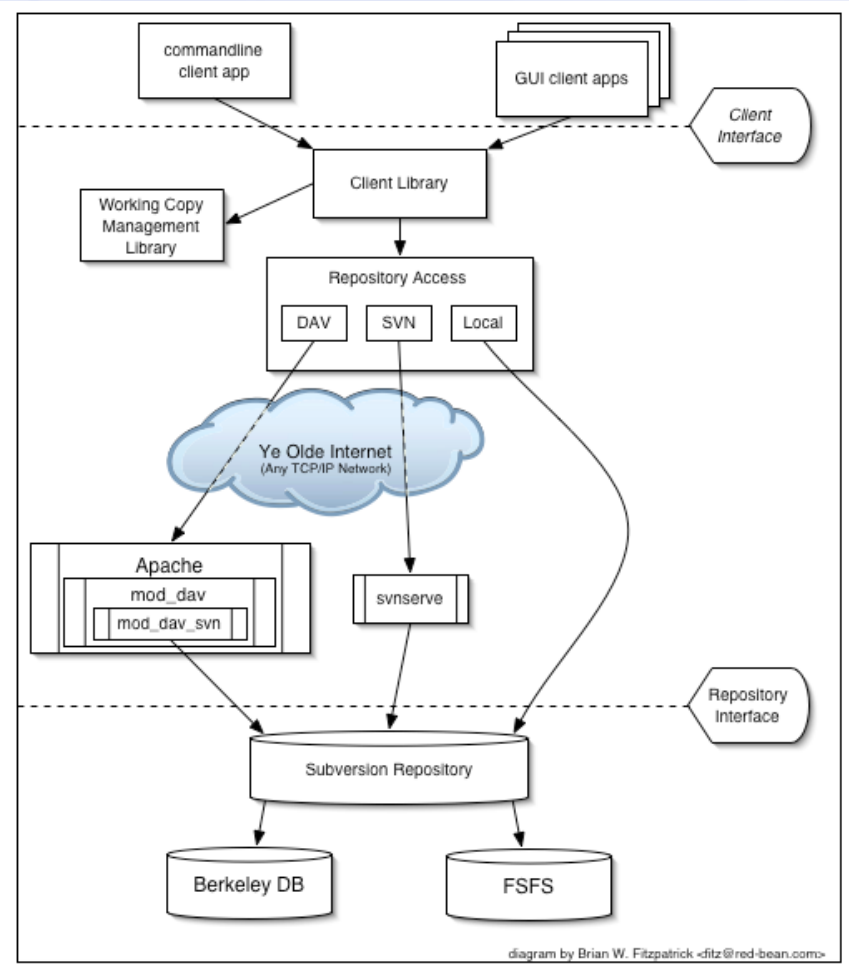
Warum Subversion?

- Einfachere Software-Entwicklung mit Gruppen
- „Undo“ von Änderungen einfach möglich
- Subversion ist weit verbreitet und kostenlos
- Kontinuierliche Entwicklung seit 2000
- Nicht wirklich in Konkurrenz zu Systemen wie Mercurial oder Git, vielmehr andere Sichtweise

CVS vs. Subversion

- Vor Subversion war das „Concurrent Versioning System“ CVS lange Zeit dominierend.
- Subversion behebt einige bekannte Design-Probleme von CVS
  (vgl. „Subversion book“)
- Gilt allgemein als „Nachfolger“ von CVS

Subversion Architektur



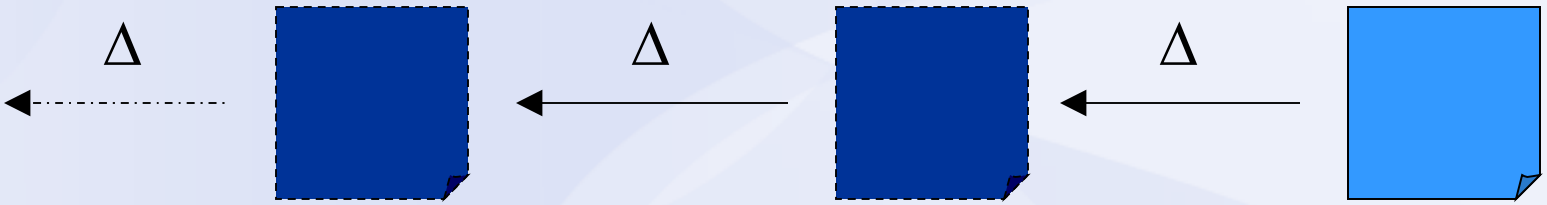
Repository vs. Arbeits-Kopie I

- Das verwaltete Projekt wird auf einer zentralen Server-DB gespeichert, dem sog. “Repository.”
- Jeder Entwickler macht ein “check out” (eine spezielle Kopie) dieses Projektes in seine lokale Arbeitsumgebung.
- Diese Kopien heißen „Arbeits-Kopien“

Repository vs. Arbeits-Kopie II

- Nach den Änderungen des Entwicklers an der Arbeits-Kopie kann er diese Änderungen als “commit” dem Repository senden
- Die anderen Entwickler erhalten diese Änderungen, in dem sie ein “update” ihrer Arbeits-Kopien machen

Differentielle Verwaltung



Vorgänger der Vorherigen Version

Vorherige Version

Aktuelle Version

Atomarität von „commits“

- Eine Menge von Veränderungen wird entweder komplett in das Repository übernommen oder gar nicht.
- Also: Entweder lassen sich alle Änderungen „committen“ oder keine.
- Daher sind teilweise Rollbacks nötig

Eigenschaften

- Jede Datei (und jedes Verzeichnis) haben assoziierte Eigenschaften
- So können z.B. manche Dateien von der Versionskontrolle ausgenommen werden, Bsp.:
  svn:ignore → “unversioned.”
- Diese werden im .svn Verzeichnis der jeweiligen Arbeitskopie abgelegt

Binäre vs. Text-Dateien

- Subversion benutzt einen binären Algorithmus zur Feststellung von Unterschieden
- Identisches Verhalten für Binär und Test-Dateien.
- Alle Daten werden komprimiert abgespeichert
- CVS behandelt beide unterschiedlich
- Dennoch: Nicht alle Binärdateien sollten in das SVN!

Ein typisches Problem...

- Benutzer A kopiert sich Datei X
- Benutzer B kopiert sich Datei X
- Benutzer A ändert X und kopiert die neue Datei X_A auf den Server.
- Benutzer B ändert seine (ältere) Version von X, schreibt die neue Version X_B auf den Server, und ***überschreibt somit die Änderungen von A, die in X_A standen***

Lock-Modify-Unlock

- Ein Ansatz mit mehreren Problemen:
- Falls zwei Benutzer unterschiedliche Abschnitte einer Datei ändern wollen, muss einer warten.
- Nach einem Lock kann man das Unlock leicht vergessen.
- Locking alleine löst nicht die Inkompatibilität zwischen verschiedenen Dateien. (vgl. „subversion book“)

Copy-Modify-Merge

- Benutzer A holt sich seine Arbeits-Kopie von X.
- Benutzer B holt sich seine Arbeits-Kopie von X.
- Benutzer A verändert seine Arbeits-Kopie von X.
- Benutzer B verändert seine Arbeits-Kopie von X.
- A „committet“ seine Arbeits-Kopie von X zurück in das Repository.
- B versucht dies auch, scheitert aber,
  weil seine Version von X nun nicht mehr aktuell ist.
- B macht ein “update”, die Veränderungen
  von A werden mit B's Version of X verschmolzen.
- Falls A's Veränderungen zu keinem Konflikt mit B's führen, ist das Update erfolgreich. Falls nicht, wird SVN die in Konflikt stehende Datei annotieren und den Fehler melden, den B dann manuell beheben muss.
- Anschließend kann B „submitten“

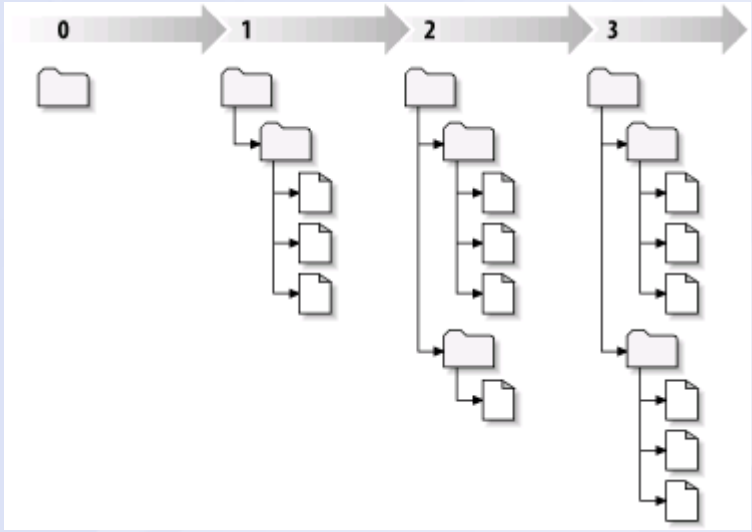
Copy-Modify-Merge in der Praxis

- Benutzer müssen nicht aufeinander warten
- In der Praxis gibt es selten Konflikte, die sich dann leit lösen lassen.
- Funktioniert nicht gut mit Binär-Daten, da diese sich schlecht verschmelzen lassen.
- Daher: Lock-Modify-Unlock zur Not ebenfalls möglich!

Revisionen

- Subversion Transaktionen sind atomar: Entweder ganz oder gar nicht
- Nach der Erstellung ist ein Repository ein leerer Ordner mit Revision 0.
- Nach dem „commit“ zu einem Repository mit Revision n, wird die Revisionsnummer auf n+1 gesetzt.

Revisionen



- Wenn wir von „Version 3 von foo.rkt“ reden, meinen wir eigentlich „foo.rkt“, wie es in Revision 3 vorliegt.

Revisionen und Arbeits-Kopien I

- Arbeits-Kopien enthalten nicht immer nur einen Versionsstand
- Sie können Dateien verschiedener Revisionen enthalten!

  → Beispiel folgt!

Revisionen und Arbeits-Kopien II

- Benutzer A “checkout” Repository *repo*.

  repo/foo.rkt 3
  repo/bar.rkt 3

- Benutzer A verändert “foo.rkt” und “committet”.

  repo/foo.rkt 4
  repo/bar.rkt 3

- Benutzer B “committet” seine Veränderungen an “bar.rkt” nach einem nötigen “update”.

  repo/system.h 5
  repo/system.cpp 5

Mögliche Zustände der Arbeits-Kopie

- Unverändert und aktuell
- Arbeits-Kopie geändert und aktuell
- Arbeits-Kopie unverändert und nicht mehr aktuell
- Arbeits-Kopie geändert und nicht mehr aktuell

Update und Commit

	update	commit
Unverändert und aktuell	Tut nichts	Tut nichts
Arbeits-Kopie geändert und aktuell	Tut nichts	Schreibt Veränderungen ins Repository
Arbeits-Kopie unverändert und nicht mehr aktuell	Ersetze Datei in Arbeitskopie mit aktueller	Tut nichts
Arbeits-Kopie geändert und nicht mehr aktuell	Verschmelzen von Arbeitskopie und aktueller Version (Merge)	Operation schlägt fehl (out-of-date error)

Ende der Einführung

Vielen Dank für die Aufmerksamkeit!