Adios CVS!

Markus Schade

Agenda

- Warum wecheln
- Überblick Subversion
  - Neuerungen in Release 1.5/1.6
- Konvertierungstools
- Konvertierung mit cvs2svn/git

Gründe zum Wechseln

- **Organisatorisch**
- **Das Erstellen eines Tags/Zweigs dauert einen Arbeitstag**
- **Es gibt dafür einen eigenen Mitarbeiter**
- **Der CVS Guru geht bald in Rente**
- **Management / Community-Entscheidung**
- **örtlich verteilte Entwicklung**

Gründe zum Wechseln

- **Technisch**

- **Bessere Performance (Checkout, Diff, Tag, etc)**
- **Bessere Skalierbarkeit (siehe code.google.com)**
- **Neue Funktionen (atomic commits, merge tracking, hooks, etc.)**

- **vermutete Datenfehler / Inkonsistenzen**
  - **handeditierte RCS-Files**
  - **gelöschte Dateien aus Attic kopiert**
  - **Tags/Branches auf nicht existente Versionen**
  - **Verzeichnisse und Dateien mit gleichem Namen**
  - **sonstige “cvs admin”-Katastrophen**

Gründe zum Wechseln

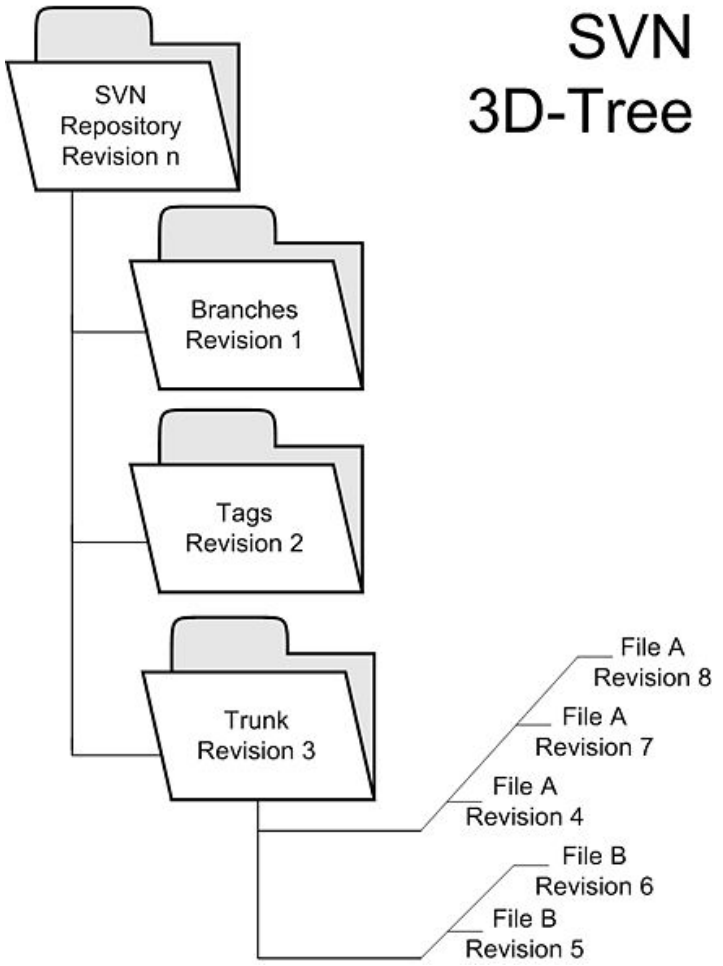
- **Sonstige**
  - **richtiges(tm) Versionskontrollsystem**
  - **Zukunftssicherheit**
  - **...**

Überblick Subversion

- erstes “next generation” Versionskontrollsystem
- ausgereift (Entwicklung seit 2000, seit 08/2001 self-hosted)
  - Git, Mercurial, Bazaar - 2005
- sehr ähnlich zu CVS, aber ohne dessen Probleme
- Highlights
  - atomische Commits
  - Versionierung von Verzeichnissen und Meta-Daten
  - Schnelles Erzeugen von Zweigen und Tags
  - partielle Checkouts
- Client/Server (zentralisiertes Modell)
  - Protokolle: file, http(s), svn, svn+ssh

Wie funktioniert Subversion

- **“3 dimensionales Dateisystem”**
- **Dateisystem mit Geschichte**
- ***trunk, branches* und *tags* sind Pfade (bzw. URLs)**
- **Die Funktion eines Pfades ist Konvention, nicht implizit**



Subversion Grundlagen

- neues Repository anlegen
  - svnadmin create myrepo
- Check out
  - svn co file://`pwd`/myrepo repo
- Layout anlegen
  - cd repo && mkdir trunk branches tags
- Daten hinzufügen
  - svn add *
- Commit / Einchecken
  - svn commit
- Synchronisation Arbeitskopie mit Repository
  - svn update
- mehr unter: http://svnbook.red-bean.com

Subversion – Überprüfen von Änderungen

- **svn status [-u]**
  - **Status der Dateien/Verzeichnisse der Arbeitskopie**
  - **ohne -u nur lokale Änderungen**
- **svn diff**
  - **Anzeige der Unterschiede zwischen Revisionen oder Pfaden**
- **svn revert**
  - **Änderungen an Dateien verwerfen**
  - **gelöschte Verzeichnisse werden nicht wiederhergestellt**

Mehrere Entwicklungszweige (Branches)

- **Tags und Zweige sind nur Pfade im Repository**
  - **svn copy -r 123 svn://repo/trunk/foo svn://repo/tags/foo-1.0**
  - **svn copy svn://repo/tags/foo-1.0 svn://repo/branches/foo-1.x-branch**
- **Immer ein Tag vor dem Zweig anlegen**
  - **nicht zwingend notwendig, aber sehr hilfreich für Merge**
- **Merge überträgt Änderungen zwischen Entwicklungszweigen**
  - **seit 1.5.x mit Änderungsverfolgung (merge tracking)**
  - **vom Hauptentwicklungszweig**
    - svn merge svn://repo/trunk
  - **vom Branch**
    - svn merge –reintegrate svn://repo/branch/foobar-branch
  - **generell**
    - svn merge URL1@REV1 URL2@REV2 WC_PATH

Weitere Neuerungen

- **Erweiterte partielle checkouts (sparse)**
  - **Leer, nur Dateien, Dateien und (leere) Verzeichnisse, vollständig**
- **Flexiblere Authentifizierung**
  - **Via Cyrus SASL (u.a. CRAM-MD5, NTLM, GSSAPI)**
- **Verbesserte Konflikterkennung und -behandlung**
  - **Interaktive Auswahl zur Lösung von Update-Konflikten**
  - **Erkennung von Konflikten in der Verzeichnisstruktur (ab 1.6)**
    - z.B. lokal modifizierte Datei vs. gelöscht im Repository

Warum Subversion

- kein Versionskontrollsystem perfekt ist
- um- und einsteigerfreundlich
- skalierbar (10++ GB Repositories)
  - DVCS (git, hg) – Aufteilung in Teilrepositories
  - z.B.: Änderung an API + Anpassungen in einem Commit möglich
- firmengeeignet(er)
  - feingranuale Autorisierung
  - Integration mit 3rd party tools (buildbot, TortoiseSVN, Subclipse, etc.)
- Basis für zukünftige Migrationen
- flexible Arbeitsweisen
  - git-svn, bzr-svn

Warum nicht Subversion

- **man kann im Flugzeug/Zug/Himalaya nicht einchecken**
- **trunk/tags/branches sind nur Konvention**
- **zentralisierte VCS sind doof!!1**

Konvertierungstools

- **cvs2svn/git**
  - **http://cvs2svn.tigris.org**
- **SVN Importer**
  - **http://www.polarion.org**
- **Tailor (generisch)**
  - **http://wiki.darcs.net/DarcsWiki/Tailor**

CVS2SVN

- existiert seit Entwicklungsbeginn von Subversion (2001)
  - Version 2.2 (November 2008)
  - benötigt Python 2 (2.4 oder besser, nicht 3.x) und GNU sort
  - Entwicklerversion (fast) immer verwendbar
- Daten und Geschichte bestmöglichst rekonstruieren
- 16 wiederholbare Teilschritte
- erlaubt gezielte Schönheitskorrekturen
- Automatisches Setzen von SVN Eigenschaften
- robust gegenüber CVS Eigenheiten/Fehlern

Konvertierungsumfang

- **kein Konvertierung / direkter Import**
  - **Einchecken eines (bereinigten, umsortierten) CVS-Exports**
- **Nur Hauptzweig**
  - **spart Speicherplatz**
  - **entspricht oft vollständiger Konvertierung bei kleinen Projekten**
- **selektiv**
  - **Ausschluß von bestimmten Symbolen (tägliche Build-Tags, Testentwicklungszweige)**
- **vollständig**
  - **default**
  - **braucht mehr Speicherplatz**
- **individuell**
  - **aufwändig**
  - **führt exakt zum gewünschten Resultat**

Vorbereitungen

- **Immer an einer Kopie des Repositories arbeiten**
- **server-seitige symbolische Links wenn möglich entfernen**
- **Text/Binärdateien korrekt markieren (empfehlenswert)**
  - **möglicher Datenverlust durch automatische Zeilenendekorrektur**
  - **Diff/Merge von Binärdateien als Text**
  - **Korrektur mit: cvs admin -kb *filename***
  - **bei gründlicher Anwendung kann *default-eol=native* genutzt werden**
  - **--eol-from-mime-type setzt svn:mime-type-Eigenschaft**

Vorbereitungen

- Auf- und Umräumen
  - kein echtes Rename/Move in SVN möglich
  - vollständiges Löschen nur durch Dump, Filter und Restore
- für nicht triviale Konvertierungen Options-Datei verwenden
  - Beispiele für SVN, Git und Mercurial vorhanden
  - cvs2svn –options=myoptions.file
  - z.B. Konvertierung mehrerer Projekte, spezielles Repo-Layout, etc
  - komplexe Transformationsregeln
- Zwischenergebnisse mit –skip-cleanup aufheben
  - Wiederholung von Zwischenschritten
  - Fortsetzen nach Abbrüchen

Symbolverarbeitung

- CVS Symbole können Zweige oder Tags sein
  - oder fälschlicherweise beides
- Verarbeitung im CollateSymbolPass (aktuell: 3. Schritt)
- Ausschluß via regulärem Ausdruck
  - --exclude='^test-.+'
- Verhalten bei mehrdeutiger Verwendung
  - --force-branch, --force-tag
  - --symbol-default=heuristic|strict|branch|tag
- Umschreiben von Symbolen
  - --symbol-transform='MUSTER:ERSATZ'
- Spezial
  - --write-symbol-info erstellt Logfile der Symbolregeln
  - Eingabe für –symbol-hints

Weitere Tipps

- CVSNT kann, muß nicht funktionieren
  - --use-cvs
- --cvs-revnums
  - speichert CVS Revisionsnummern als SVN-Eigenschaft (property)
  - verursacht Konflikte beim Mergen
- Out-of-Memory Fehler
  - durch große Changesets
  - RAM und Swap erhöhen
  - 64bit System nutzen
- Konvertierung beschleunigen
  - kleine Repos in RAM Disk
  - große Repos auf schnelles RAID0
- Viel Glück

Danke! Fragen? Antworten.