CVS, Subversion und Darcs

Agenda

- Motivation
- CVS
- Subversion
- Darcs
- Fazit
- Literatur

Motivation

CVS

Subversion

Darcs

Fazit

Literatur

Motivation

- Änderungen erzeugen Fehler, die häufig erst später erkannt werden
- Mischen der Änderungen bei mehreren Entwicklern – Gefahr des Überschreibens
- Wunsch – alle Änderungen nachvollziehen $\longrightarrow$ hohes Datenaufkommen

$\longrightarrow$ Einsatz von Versionierungssystemen

Motivation

CVS

Subversion

Darcs

Fazit

Literatur

- Dezember 1986 - Menge von Shell Scripts von *Dick Grune*
- April 1986 - Design und Coding von *Brian Berliner*, später unterstützt durch *Jeff Polk*
- wird unter GPL angeboten

Begriffe

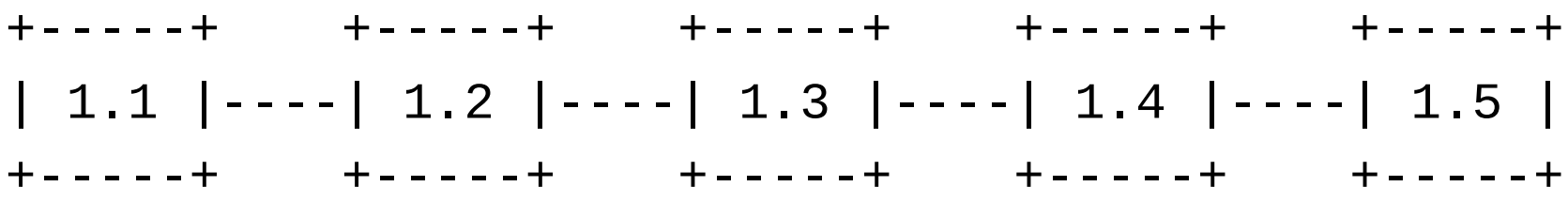
Repository enthält die Dateien, die unter Versionsverwaltung stehen

Modul Zusammenfassung von Dateien/Verzeichnissen – entspricht meist dem Projekt

Revision & Version CVS unterscheidet zwischen Revision (per-file-concept) und Version (per-package-concept)

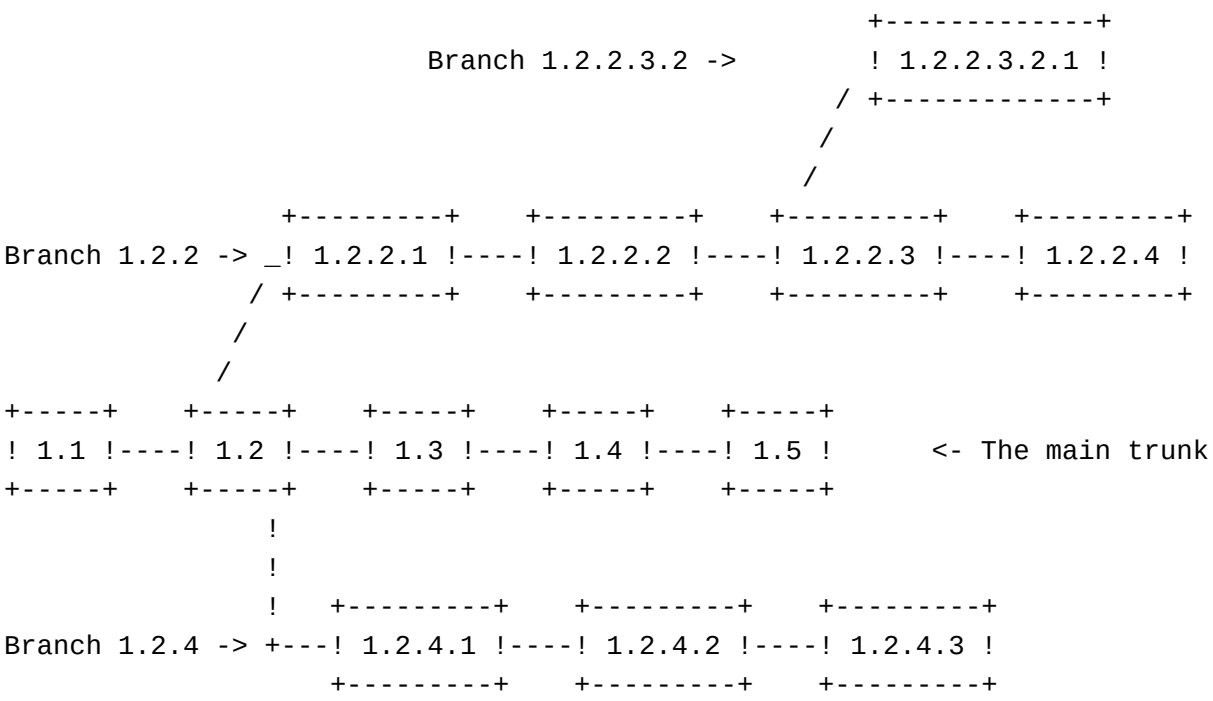
Revision

- Jede Version einer Datei hat eine eindeutige *revision number*
- Revisionsnummer – ganze, durch Punkte getrennte Zahlen
- 1.1 ist die erste Revisionsnummer einer Datei



Version

- Jede Branch hat eine eigene *branch number*
- Anhängen einer neuen Stelle an die Revisionsnummer, von der der Branch abgespalten wurde



Repository

- Enthält eine Kopie der Dateien und Verzeichnisse, die sich unter der Versionskontrolle befinden
- Dateien sind als *history files* hinterlegt
- Auf die Dateien das Repository wird nicht direkt zugegriffen
- Das Repository kann sich auf dem lokalen oder einem anderen Rechner befinden
- Repository erstellen

```
cvs -d /usr/local/cvsroot init
```

- Pfad zum Repository
  - Umgebungsvariable

```
$ export CVSROOT=/usr/local/cvsroot
```

  - -d Option der Kommandos

```
:methode:user@hostname:/pfad/zu/dem/repository
```

Modul erstellen

```
$ cd projektverzeichnis
$ cvs import -m "first project" hello new start
```

- Modul *hello* wird im Repository erzeugt
- Dateien und Verzeichnisse des Projektes werden im Repository erstellt

```
/work
|
+--cvsroot
|  |
|  +--CVSROOT
   |
   +--hello
```

Lokale Kopie aus dem Repository holen

```
$ cvs checkout hello
```

- Ein lokales Verzeichnis *hello* wird erstellt und mit den Dateien und Verzeichnissen des Moduls gefüllt

Dateien hinzufügen/entfernen

- Hinzufügen

```
$ cvs add datei
```

- Entfernen

```
$ cvs remove datei
```

- Die Operation wird erst bei dem nächsten commit wirksam

Änderungen übernehmen

```
$ cvs commit -m "kurze Beschreibung der "Anderung"
```

- Wird die -m Option weggelassen, wird der Editor ($EDITOR) gestartet
- Lognachrichten sollten die Änderung kurz beschreiben
- Auslesen der Lognachrichten mit

```
$ cvs log datei
----------------------------
revision 1.1
date: 2004/05/29 13:23:54;  author: rgion;  state: Exp;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2004/05/29 13:23:54;  author: rgion;  state: Exp;  line
importierte sourcen
```

Abgleich mit dem Repository

```
$ cvs update
```

- Die Änderungen des Repository werden in die lokalen Dateien übernommen
- Geänderte Dateien werden aufgelistet

  U - updated - Die Änderungen des Repository wurden in die lokale Datei, die unverändert war, übernommen

  M - merged - Die Datei wurde lokal und im Repository geändert - die Änderungen wurden in der lokalen Datei gemischt

  C - conflict - Die Änderungen konnten nicht gemischt werden - manueller Eingriff erforderlich

  ? - unknown - Über die Datei ist nichts bekannt - meistens nocht nicht hinzugefügt

Branch

- Branch erstellen

```
$ cvs rtag -b -r branchname module
```

  - Ein Branch wurde im Repository erstellt

- Branch auschecken

```
$ cvs checkout -r branchname module
```

  - Änderngen betreffen jetzt nur den Branch

- Branch mit aktueller Arbeitskopie mischen

```
$ cvs update -r branchname
```

Tagging

```
$ cvs tag version_1_0
```

- Symbolische Bezeichnung für Dateien in einer bestimmten Revision
- Version auschecken

```
$ cvs checkout -r version_1_0 module
```

Keyword Expansion

- Bestimmte Schlüsselwörter werden automatisch durch CVS ersetzt

  Id - Dateiname, Revision, Datum, Autor, Status

  $Date$ - Datum und Zeit des commits

  $Autor$ - Autor der Datei

  $Revision$ - Revision der Datei

  Log - Log Einträge

```
/**
 * Der <code>DataHandler</code> dient als Verbindung zur Datenbank.
 *
 * @author Roland Gion
 * @version $Id: DataHandler.java,v 1.51 2005/05/09 07:32:45 rgion Exp $
 */
class DataHandler {
```

Schwächen

- Entspricht nicht mehr den Anforderungen an Versionierungssystem

  Skalierbarkeit - viele Dateien oder schlechte Verbindung $\longrightarrow$ langsame Operationen

  Zugriff - Zugriff auf *rsh* oder *ssh* beschränkt

  Binärdaten - Änderungen werden vollständig gespeichert $\longrightarrow$ hoher Speicherplatzbedarf

  Historie von Verzeichnissen - Verzeichnisse nicht in der Versionsverwaltung

  keine Transaktionskontrolle - commit-Operationen können zu inkonsistentem Repository führen

  Reorganisation - Umbenennen/Verschieben von Dateien problematisch - im Repository operieren um die Historie zu erhalten

Graphische Clients

WinCVS - häufig genutzt - ausgereift

TortoiseCVS - wird in den Windows-Exporer eingebunden - Steuerung über Kontextmenü

Cervisia - für KDE Nutzer

In einigen IDEs, wie z.B. Eclipse bereits vorhanden

Motivation

CVS

Subversion

Darcs

Fazit

Literatur

Subversion

- *CollabNet* ist Hersteller von *SourceCast* - beinhaltet ein Versionsverwaltungssystem (CVS)
- 2000 - Beginn CVS durch neues System (Subversion) abzulösen
- Nach 14 Monaten Entwicklungszeit – Subversion hostet sich selbst
- look-and-feel CVS nachempfunden
- de-facto Nachfolger von CVS
- Neuentwicklung - Beseitigung der Schwächen von CVS
- Unter Apache License angeboten
- Bezug http://subversion.tigris.org
- Unterstützte OS: UNIX, Linux, Win32, BeOS, OS/2 und MacOS X

Repository

- Keine *flat files* mehr – Datenbank (Berkeley DB)
  - Transaktionssicherheit - commit ist atomar
  - *redo-logs* werden mitgeschrieben – `svnadmin recover`
- *hot-backup* (im laufenden Betrieb) – `svnadmin hotcopy`
- Revisionsnummer pro Repository nicht pro Datei
- Repository anlegen

```
$ svnadmin create /usr/local/svnroot
```

Vorteile

- Keine Unterscheidung zwischen Dateien und Verzeichnissen
- Umbenennen von Verzeichnissen/Dateien
- Keine Unterscheidung zwischen Text und Binärdateien – *binary diff*
- Einfacheres Branchen
- Protokolle
  - file:// – lokales Filesystem
  - svn:// – standalone SVN-Server
  - svn+ssh:// – SVN-Serer mit ssh-Tunnel
  - http:// – Apache2 und WebDAV
  - https:// – Apache2 und WebDAV und SSL-Verschlüsselung
- Bessere Rechteverwaltung (Apache2)

Vergleich CVS- und Subversion-Kommandos

Aktion	CVS	SVN
Initialisierung	cvs init	svnadmin create directory
Checkout	cvs checkout module	avnadmin checkout method://rep/path
Dateien hinzufügen	cvs add file	svn add file
Dateien entfernen	cvs delete file	svn delete file
Commit	cvs commit file[s]	svn commit file[s]
Logs ansehen	cvs log file[s]	svn log file[s]

Projekt anlegen

```
$ mkdir hello
$ cd hello
$ mkdir trunk tags branches
$ svn import . file:///pfad/zu/dem/repository -m "initial impor
```

```
/work
|
+--hello
|  |
|  +--trunk
|  |
|  +--branches
|  |
|  +--tags
|
+--...
```

Branching & Tagging

```
$ svn copy http://svn.beispiel.com/repos/hallo/trunk \
 http://svn.beispiel.com/repos/hallo/tags/release-1.0 \
 -m "Release 1.0 des Projekts hallo"
```

- Branching und Tagging durch kopieren in das jeweilige Verzeichnis
- Kopien mittels *cheap copy* erstellt - nur als kopiert markiert

Checkout

- Wird beim checkout kein Modul angegeben wird das gesamte Repository ausgecheckt
- Lokale Kopie ermöglicht offline-Betrieb – Reduktion der Netzlast

Grapische Clients

TortoiseSVN - Pendant zu TortoiseCVS

RapidSVN - nicht einfach zu bedienen – für viele UNIXe verfügbar

subeclipse - Eclipse Plugin

Motivation

CVS

Subversion

Darcs

Fazit

Literatur

Darcs

- David’s Advanced Revision Control System
- Entwickler: David Roundy
- Unter GPL lizensiert
- In Haskell geschrieben
- Bezug http://www.darcs.net
- Unterstützte OS: Linux, Win32, BSD und MacOS X

Repository

- Jede lokale Kopie ist ein vollständiges Repository
  - Vorteile
    - Ermöglicht mobile Entwicklung
    - Reduziert Netzlast
  - Nachteil
    - Größeres lokales Repository
- Mehrere Repsository können sich Verzeichnisse teilen, wenn sie sich auf dem gleichen Filesystem befinden
- Repository anlegen
  - Das Verzeichnis `_darcs` für die Verwaltungsdaten wird in dem Projektverzeichnis angelegt

```
$ mkdir hallo
$ cd hallo
$ darcs initialize
```

Vorteile

- Änderungen (Patches) werden gespeichert
- Abgleich von Änderungen zwischen verschiedenen Repository
- Interaktiv
- Patches können per Mail versendet werden

CVS- und Darcs-Kommandos

Aktion	CVS	SVN
Initialisierung	cvs init	darcs initialize
Checkout	cvs checkout	darcs get
Update	cvs update	darcs whatsnew — darcs pull
Dateien hinzufügen	cvs add	darcs add
Dateien entfernen	cvs delete	darcs delete
Commit	cvs commit	darcs record — darcs push
Änderungen bestimmen	cvs diff	darcs diff — darcs whatsnew
Branch erzeugen	cvs tag -b	darcs get
Version erzeugen	cvs tag	darcs tag

Projekt anlegen

- Falls das Projekt neu angelegt wird

```
$ darcs add datei[en]
```

- Falls das Projekt aus einem anderen Repository ausgelesen wird

```
$ darcs get http://darcs.beispiel.com/repos/hallo
```

Branching & Tagging

- Beides nur durch Anlegen eines neuen Repository zu realisieren

Projektverzeichnis

```
\
...
|
+--hallo--------------test.c
   |
   +--_darcs----------inventory
      |
      +--current------test.c
      |
      +--inventories
      |
      +--patches
      |
      +--prefs
```

Inventory-Datei

```
[initial add
**20050515175837]
[changed searchname
**20050515180841]
[changed varname
**20050515181519]
[TAG v0.1
**20050515182916]
[change define
**20050515210859]
```

Erster Patch

```
[initial add
**20050515175837] {
addfile ./testfile1.c
hunk ./testfile1.c 1
+#include <stdio.h>
+#include <stdlib.h>
+
+#define name "Urbansky"
+
+struct element {
+       struct element *next;
+       char *zeichenkette;
+};
+
+void main()
```

Fazit

- CVS
  - Immer noch das gebräuchlichste System
  - Große Projekte wie sourceforge.org setzen noch auf CVS
  - Ist nicht mehr *state of the art*
- Subversion
  - Dank CVS *look and feel* Potential um CVS abzulösen
  - Projekte wie Apache setzen bereits darauf
  - Viele Pluspunkte bei Sicherheit (Repository)
  - Binaries für viele OS erleichtert Durchbruch
  - Gute graphische Clients
- Darcs
  - Ebenfalls CVS ähnliches *look and feel*
  - Sehr gut für mobile Entwicklung geeignet
  - Leider keine graphischen Clients

Motivation

CVS

Subversion

Darcs

Fazit

Literatur

Literatur

CVS http://www.cvshome.org

Subversion http://subversion.tigris.org

Darcs http://www.darcs.net

Ende

powerd by LaTeX 2ε

Zuletzt geändert am 17. Mai 2005.