CVS

19.11.2002

Daniel Weber & Zhen Zhang

Betreuer: Volker Baier

Was ist CVS? Was leistet CVS?

- ermöglicht Revisionsverwaltung für Projekte
- arbeitet auf ganzen Verzeichnisbäumen
- zentrales Archiv im Netzwerk möglich
- erleichtert Koordination der Arbeit

Anwendungsbereiche

- Softwareprojekte
  - viele Open-Source-Projekte
- Dokumentationen
- Webseiten

- Einzelprojekte
  - Archivierung
- Projektteams
  - gleichzeitiges Arbeiten
  - Teleworking

Bestandteile von CVS

- Repository
  - Pfad zum Repository mit -d angeben oder die Environmentvariable CVSROOT setzen
  - bei Zugriffstyp „ext“ die Loginshell (z.B.: SSH) über die Environmentvariable CVS_RSH setzen
- Projekte im Repository
- lokale Arbeitskopien

```
cvs -d /home/cvsroot ...                          ; lokal
cvs -d:pserver:user@server:/home/cvsroot ...      ; CVS Server
cvs -d:ext:user@server:/home/cvsroot ...          ; CVS Server per z.B. RSH
```

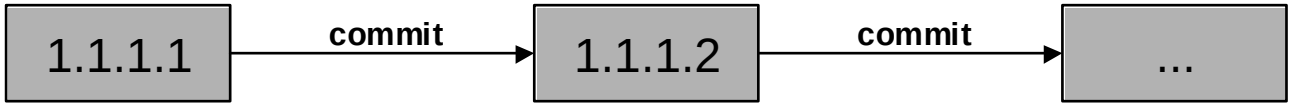
Repository

- befindet sich unter $CVSROOT
- enthält alle Dateien und Verzeichnisse, die unter Versionskontrolle stehen
- aufgeteilt in zwei Bereiche:
  - CVSROOT
  - weitere Verzeichnisse (für einzelne Projekte)
- muss mit „init“ angelegt werden

```
export CVSROOT=/home/cvsroot
cvs init
        ... oder ...
cvs -d /home/cvsroot init
```

Revisionen, Releases und Versionen

- Revisionen: Zwischenversionen, die im Laufe der Programmentwicklung permanent auftreten. Übernimmt CVS selbstständig.



- Releases: Fertige Versionen, die getestet sind und an den Kunden weitergegeben werden. Können mittels Tags markiert werden.
- Wichtig: Revisionsnummern sind meist für jede Datei individuell.

Typische Aufgaben

- import: Importieren eines neuen Projekts ins Repository
- checkout: Anlegen einer lokalen Arbeitskopie.
- update: Abrufen von Neuerungen aus dem Repository
- add/remove: Hinzufügen und Entfernen von Projektdateien
- commit: Einspielen von Änderungen in das Repository

import

- erstellt ein neues Modul im Repository
- Projekt muß bereits als Verzeichnisbaum vorliegen
- kopiert die Projektdateien als Urfassung (meist Revision 1.1.1.1) in das Repository
- Arbeitsverzeichnis danach löschen und mit „checkout“ neu angelegen

```
cd TolleSoftware
cvs import TolleSoftware TolleFirma Start
```

checkout

Erstellt eine lokale Arbeitskopie eines Projekts

- legt das Verzeichnis mit dem Modulnamen an
- kopiert höchste Revision aller Projektdateien ins Arbeitsverzeichnis
- alternativ: bestimmte Revision auswählbar

```
cvs checkout TolleSoftware
```

update

- überprüft lokale Arbeitskopie und Modul im Repository auf Unterschiede
- aktualisiert Arbeitskopie ggf. auf den aktuellsten Stand **aus dem Repository**.
- behebt ober markiert mögliche Konflikte in der lokalen Arbeitskopie.
- weist auf lokale Änderungen hin, spielt diese aber **nicht in das Repository**.

```
cvs update
```

add / remove

- dient zum Hinzufügen neuer oder Löschen alter Projektdateien
- Dateien müssen vorher erstellt oder gelöscht werden
- protokolliert den Befehl, setzt ihn aber noch **nicht** im Repository um
- wird erst nach „commit“ wirksam

```
touch newclass.java
cvs add newclass.java
cvs commit
```

commit

- überträgt lokale Änderungen und neue Dateien nun tatsächlich ins Repository bzw. entfernt gelöschte Dateien ab aktueller Revision
- erhöht dabei den Revisionszähler geänderter Dateien bzw. setzt den Revisionszähler neuer Dateien
- Anwender kann Änderungen durch Logmessage kommentieren

```
vi newclass.java          ; etwas ändern
cvs commit
```

Weitere Aufgaben

- diff: Unterschiede zwischen Revisionen ermitteln
- release: Freigeben der lokalen Arbeitskopie
- tag: Revision einen Namen zuweisen

diff

Ermittelt die Unterschiede

- zwischen Arbeitskopie und bestimmter Revision
- zwischen Revisionen im Repository

```
cvs diff -r 1.3.2.1 Matrix.java            ; Arbeitskopie und Revision
         ... oder ...
cvs diff -r 1.3.2.1 -r 1.4 Matrix.java     ; zwei Revisionen
```

release

- gibt lokale Arbeitskopie frei
- warnt bei Änderungen, die noch nicht mit „commit“ ins Repository überspielt wurden
- löscht ggf. Arbeitskopie (-d Option)

```
cd ..
cvs release -d TolleSoftware
```

tag

Weist einer bestimmten Revision einen Namen zu

- meist unterschiedliche Revisionen bei Dateien $\Rightarrow$ mit Tags kann man gemeinsamen Zwischenstand markieren
- Tags können auch beim „checkout“ zum abrufen eines bestimmten Zwischenstands genutzt werden

```
cvs tag beta-1-2
...
cvs checkout -r beta-1-2
```

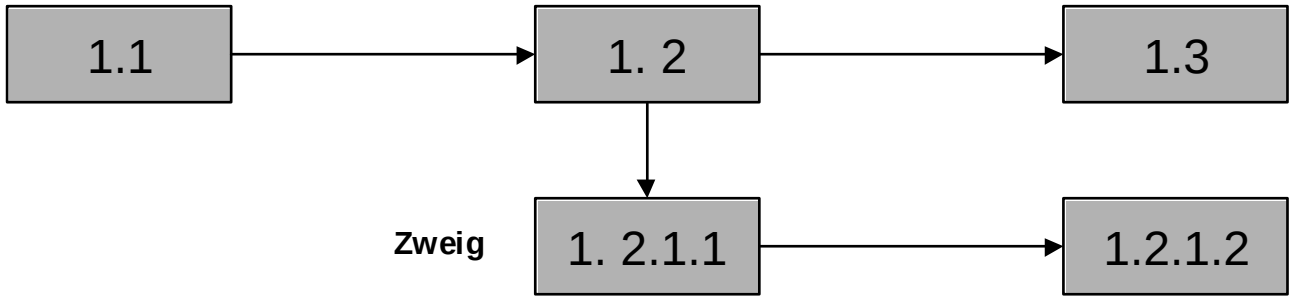
Zweige (Branches)

Nützlich wenn neben der Entwicklung der nächsten Version parallel am Debugging der aktuellen Version gearbeitet werden soll.

- Anlegen eines Zweiges
- Zweige wiedervereinigen

Zweige anlegen

- erstellt aus der lokalen Arbeitskopie einen neuen Zweig im Repository
- „checkout“ und „update“ können auf Zweige ausgeführt werden




```
cvs tag -b BugfixZweig
...
cvs update -r BugfixZweig
```

Zweige vereinigen

- alle Änderungen, die in einem anderen Zweig stattgefunden haben, werden in die Arbeitskopie übernommen
- mit folgendem „commit“ sind die Änderungen im auch bisherigen Zweig der Arbeitskopie enthalten

```
cvs checkout TolleSoftware
cd TolleSoftware
cvs update -j BugfixZweig
cvs commit
```

Keywords

- Schlüsselwörter werden in $-Zeichen eingeschlossen
- werden bei „checkout“ und „update“ expandiert, d.h. mit Werten belegt
- muss bei binären Dateien abgeschaltet werden (-kb)

```
$Author$
...
$Author: zhen $
```

grafische Frontends

- Emacs
- Sun ONE (früher Forté)
- Pharmacy
- CvsGui: WinCVS, MacCVS, gCVS
- jCVS, SmartCVS
- TkCVS