Concurrent Versions System (CVS)

Siarhei Trushyn

Proseminar Unix Tools
TUM

22. November 2005

Gliederung

1. Einführung
2. Archiv
3. Projekt mit CVS
4. Momentaufnahmen
5. Zusammenfassung

Was ist CVS?

Definition

Concurrent Versions System (CVS) bezeichnet ein Programm zur Versionsverwaltung von Dateien, hauptsächlich Software-Quellcode und ist eine Weiterentwicklung von RCS

So begann alles...

1986 - Dick Grune schreibt CVS Shell Skripte
1989 - Brian Berliner und Jeff Polk entwickeln die CVS Software. Viele Aspekte von Dick Grune werden übernommen

Was bietet CVS und was nicht?

Das bietet CVS ...

- Versionskontrollverwaltung
- Gemeinsame Entwicklung von vielen Personen
- Speicherplatzersparnis
- Open Source (GPL-Lizenz)

... und das nicht!

- kein Entwicklungssystem
- kein Ersatz für Projektleitung
- kein Ersatz für Kommunikation zwischen Entwicklern
- keine systematische Behebung von Fehlern
- keine automatisierte Verifiezierungssoftware
- keine Prozessmodellierung

Beispiel

Aufgabe

Aus einem vorhandenen Projekt soll eine Datei geändert werden

Lösung

```
> cvs checkout projekt_01
> cd projekt_01
> vi main.c
> cvs commit -m 'Fehler korrigiert'
> cd ..
> cvs release -d projekt01
```

Beispiel

Aufgabe

Aus einem vorhandenen Projekt soll eine Datei geändert werden

Lösung

```
> cvs checkout projekt_01
> cd projekt_01
> vi main.c
> cvs commit -m 'Fehler korrigiert'
> cd ..
> cvs release -d projekt01
```

Beispiel

Aufgabe

Aus einem vorhandenen Projekt soll eine Datei geändert werden

Lösung

```
> cvs checkout projekt_01
> cd projekt_01
> vi main.c
> cvs commit -m 'Fehler korrigiert'
> cd ..
> cvs release -d projekt01
```

Beispiel

Aufgabe

Aus einem vorhandenen Projekt soll eine Datei geändert werden

Lösung

```
> cvs checkout projekt_01
> cd projekt_01
> vi main.c
> cvs commit -m 'Fehler korrigiert'
> cd ..
> cvs release -d projekt01
```

Beispiel

Aufgabe

Aus einem vorhandenen Projekt soll eine Datei geändert werden

Lösung

```
> cvs checkout projekt_01
> cd projekt_01
> vi main.c
> cvs commit -m 'Fehler korrigiert'
> cd ..
> cvs release -d projekt01
```

Beispiel

Aufgabe

Aus einem vorhandenen Projekt soll eine Datei geändert werden

Lösung

```
> cvs checkout projekt_01
> cd projekt_01
> vi main.c
> cvs commit -m 'Fehler korrigiert'
> cd ..
> cvs release -d projekt01
```

Beispiel

Aufgabe

Aus einem vorhandenen Projekt soll eine Datei geändert werden

Lösung

```
> cvs checkout projekt_01
> cd projekt_01
> vi main.c
> cvs commit -m 'Fehler korrigiert'
> cd ..
> cvs release -d projekt01
```

Fertig

Was ist ein CVS Archiv?

Definition

CVS Archiv oder auch Repository verwaltet komplette Kopien von Dateien und Verzeichnissen, die sich unter einer Versionskontrolle befinden

Benutzung des Archivs

Falsche Benutzung des Archivs

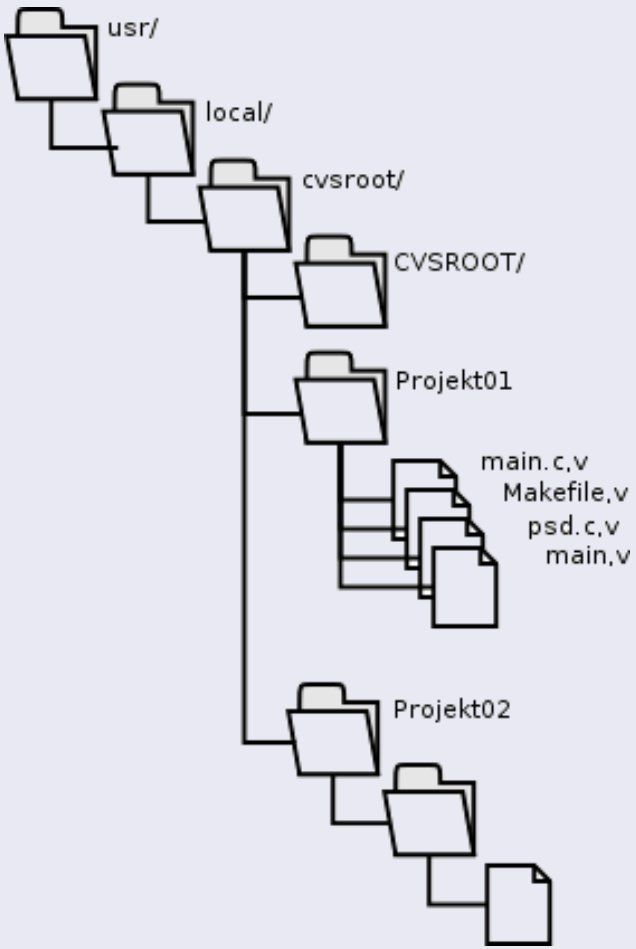
Ein CVS Archiv sollte nur in Ausnahmefällen von Hand geändert oder manipuliert werden (z.B. Fehler im Archiv)
Ausnahmen: Umbenennen/Verschieben von Dateien/Verzeichnissen und CVS administrationsbezogene Einstellungen

Richtige Benutzung des Archivs

Es sollen CVS Befehle verwendet werden

Aufbau eines CVS Archivs

Veranschaulichung



Aufbau

- CVSROOT enthält Administrationsdateien für das ganze Archiv
- Logischer Aufbau des Archivs gleich dem Arbeitsverzeichnis
- Alle Änderungen und Log-Informationen in den jeweiligen ,v Dateien festgehalten
- Alle Änderungen der Dateien sind reproduzierbar!

Anlegen eines Archivs

Vorgehensweise

```
> cvs -d /usr/local/cvsroot init
> ls /usr/local/cvsroot
CVSROOT
```

Bemerkung

Für Benutzung eines lokalen Archivs ist nichts weiter notwendig. Soll ein CVS Archiv auf einem entfernten Rechner eingerichtet werden, muss dieser zusätzlich konfiguriert werden, damit er auf CVS Anfragen antwortet.

Zugriff auf ein Archiv

Allgemein

```
export CVSROOT=<Archiv> # oder ...
cvs -d <Archiv> <Kommando>
```

Lokales Archiv

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

Entferntes Archiv - PSERVER

```
> export CVSROOT=:pserver:alex@srv:/usr/local/cvsroot
> cvs login #nur einmalig durchzufuehren
```

Entferntes Archiv - SSH

```
> export CVSROOT=:ext:alex@srv:/usr/local/cvsroot
> export CVS_RSH=ssh
```

Übernahme eines vorhandenen Projekts

Vorgehensweise

```
> cd projekt
> cvs import -m "Projekt01" projekt01 unix-tools start
N projekt01/Makefile
N projekt01/main.c
No conflicts created by this import
```

Hinweis

- CVS erzeugt im Archiv ein Modul projekt01
- Alle Dateien und Verzeichnisse im Verzeichnis projekt werden rekursiv in das Modul übernommen
- unix-tools und start bezeichnen Tags für das Projekt

Erzeugen eines neuen Projekts

Vorgehensweise

```
> mkdir projekt01
> touch projekt01/Makefile
> touch projekt01/main.c
> cd projekt01
> cvs import -m "Projekt01" projekt01 unix-tools start
```

Hinweis

- CVS erzeugt im Archiv ein Modul projekt01
- Alle Dateien und Verzeichnisse im Verzeichnis projekt werden rekursiv in das Modul übernommen
- unix-tools und start bezeichnen Tags für das Projekt

Auschecken einer Arbeitskopie

Vorgehensweise

```
> cvs checkout projekt01
cvs checkout: Updating projekt01
U projekt01/Makefile
U projekt01/main.c
> cd projekt01
> ls
CVS Makefile main.c
```

Hinweis

- checkout Befehl holt aus dem Archiv die aktuelle Version des Projekts in das Arbeitsverzeichnis
- Hierbei handelt es sich um eine Kopie des Projekts, das im CVS Archiv abgespeichert ist

Änderungen anschauen - Update

Vorgehensweise

```
> cvs -n update
cvs update: Updating .
M main.c
```

Hinweis

- Update vergleicht den Gesamtzustand der Arbeitskopie mit dem Zustand des Projekts im Archiv
- Somit kann man feststellen, was man selber und was andere gemacht haben
- M bedeutet: Datei geändert und kein commit durchgeführt
- Globale Option -n verhindert jegliche schreibende Operationen

Änderungen anschauen - Diff

Vorgehensweise

```
> cvs diff main.c
cvs diff: Diffing main.c
Index: main.c
===================================================================
RCS file: /usr/local/cvsroot/projekt01/main.c,v
retrieving revision 1.1
diff -r1.1 main.c
4c4
< printf("hello wold!");
---
> printf("hello world!");
```

Änderungen anschauen - Diff

Hinweis

- Diff vergleicht die möglicherweise veränderten Dateien der Arbeitskopie mit den entsprechenden Gegenstücken aus dem Archiv
- Mit der Option -c wird das Diff-Format etwas leserlicher dargestellt

Änderungen speichern - Commit

Vorgehensweise

```
> cvs commit -m "Fehler behoben"
cvs commit: Examining .
/usr/local/cvsroot/projekt01/main.c,v <-- main.c
new revision: 1.2; previous revision: 1.1
```

Hinweis

- commit Befehl schickt alle Änderungen an das Archiv
- Wird keine Datei angegeben, wird das aktuelle Verzeichnis rekursiv nach Änderungen durchgesucht
- Mit -m "Kommentar" wird eine Beschreibung für die Änderung angegeben

Mehrere Benutzer

Problem

```
> cvs commit
cvs commit: Examining .
cvs commit: Up-to-date check failed for ‘main.c’
[...]
```

Hinweis

- Commit schlägt fehl, falls andere Benutzer gleiche Dateien verändert haben
- In diesem Fall müssen die unterschiedlichen Revisionen zusammengeführt werden

Was haben andere gemacht?

```
> cvs -n update
```

Revisionen zusammenführen

Vorgehensweise

```
> cvs update
cvs update: Updating .
RCS file: /usr/local/cvsroot/projekt01/main.c,v
retrieving revision 1.3
retrieving revision 1.4
Merging differences between 1.3 and 1.4 into main.c
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in main.c
C main.c
```

Hinweis

Update vereinigt die Datei im Arbeitsverzeichnis und die aus dem Archiv und aktualisiert die Revisionsnummer

Konflikte lösen

Hinweis

Treten Konflikte auf, müssen diese vom Benutzer durch Editieren der betroffenen Datei gelöst werden

Vorgehensweise

```
[...]
<<<<<<< main.c
for (i = 0; i < size; i++)
=======
for (i = 0; i < size - 1; i++)
>>>>>>> 1.4
[...]
```

Nicht vergessen!

```
cvs commit
```

Überblick behalten - Log

Vorgehensweise

```
> cvs log main.c
RCS file: /usr/local/cvsroot/projekt01/main.c,v
Working file: main.c
[...]
----------------------------
revision 1.1
date: 2005-10-20 08:40:07 +0000;  author: set; [...]
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2005-10-20 08:40:07 +0000; [...]  lines: +0 -0
Projekt01
```

Überblick behalten - Log

Hinweis

- Log liefert einen Überblick über die Geschichte der Dateien
- Damit ist es eine gute Möglichkeit zu erfahren, was sich mit einzelnen Dateien in der Geschichte des Projekts abgespielt hat
- Wird keine Datei als Parameter angegeben werden rekursiv alle Dateien angezeigt

Änderungen zurücknehmen

Ausgangssituation

```
revision 1.5
date: 1999/04/20 04:14:37; [...] lines: +1 -1
adjusted middle line
```

Änderungen anschauen

```
cvs diff -c -r 1.4 -r 1.5 hello.c
Index: main.c
===================================================
RCS file: /usr/local/cvsroot/projekt01/main.c,v
retrieving revision 1.4
retrieving revision 1.5
diff -r1.4 -r1.5
> printf("size: %d\n", size);
```

Änderungen zurücknehmen

Hinweis

- Die Reihenfolge der Revisionen sollte chronologisch angegeben werden
- Wird nur eine Revision angegeben, so wird mit der aktuell ausgecheckten verglichen
- Beim Undo geht die zurückgenommene Revision nicht verloren, sie wird mehr ein teil der Historie und kann jederzeit wiederhergestellt werden

Revisionsabfolge

[1.4] -> 1.5 -> [1.6]

Änderungen zurücknehmen - Schnelle Methode

Vorgehensweise

```
> cvs update
> cvs update -j 1.5 -j 1.4 main.c
> cvs commit
```

Hinweis

- Mit der Option j (Join) ermittelt CVS Unterschiede zwischen den zwei angegebenen Revisionen und wendet sie auf die Arbeitskopie an
- Die Reihenfolge der Revisionen ist extrem wichtig!

Aufräumen

Akzeptable Methode

```
> rm -r projekt01
```

Schönere Methode

```
> cvs release -d projekt01
[...]
```

Hinweis

- release Befehl überprüft, ob noch ungespeicherte Änderungen vorhanden sind und protokolliert diesen Vorgang
- Mit der Option -d wird das Verzeichnis ebenfalls gelöscht

Andere nützliche Befehle

Dateien hinzufügen

```
> cvs add main.c
> cvs commit
```

Dateien entfernen

```
> rm main.c
> cvs remove main.c
> cvs commit
```

Verzeichnisse hinzufügen

```
> cvs add bin
```

Binärdateien

```
> cvs add -kb main
> cvs commit
```

Einen Zeitpunkt markieren

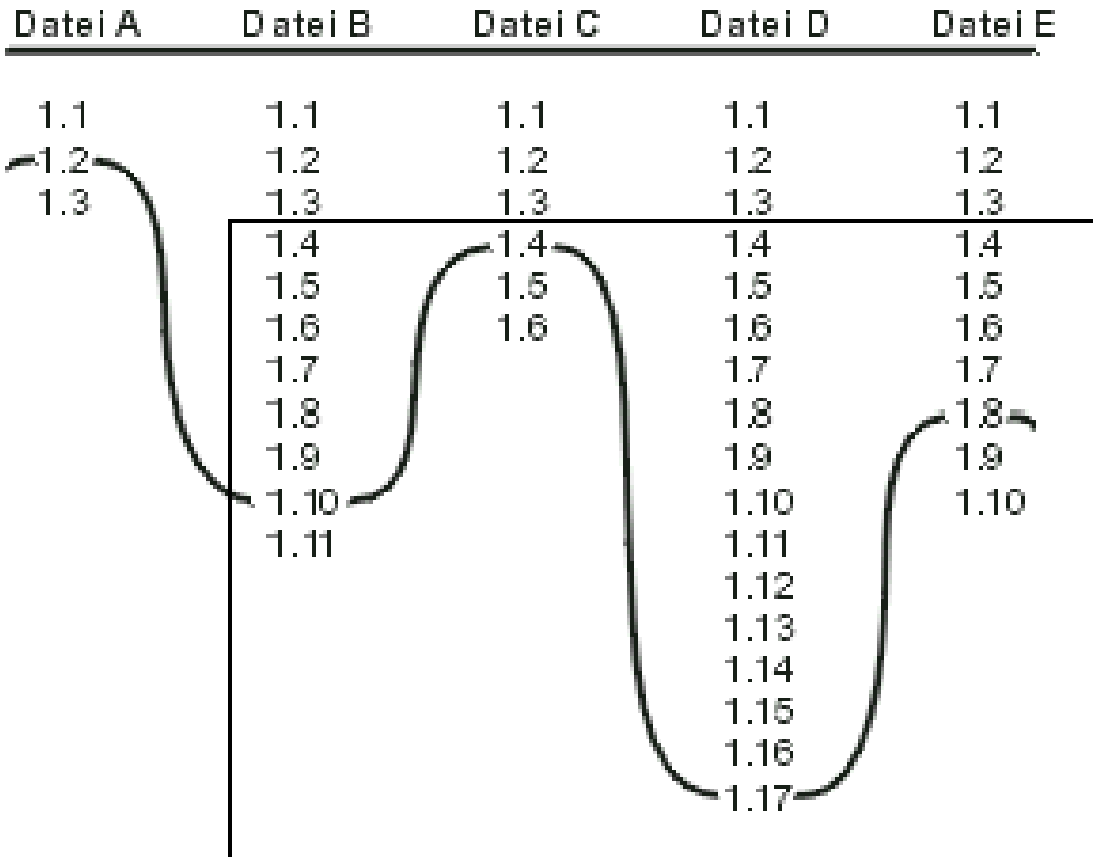
Vorgehensweise

```
> cvs tag rel-01
cvs tag: Tagging .
T Makefile
T main.c
```

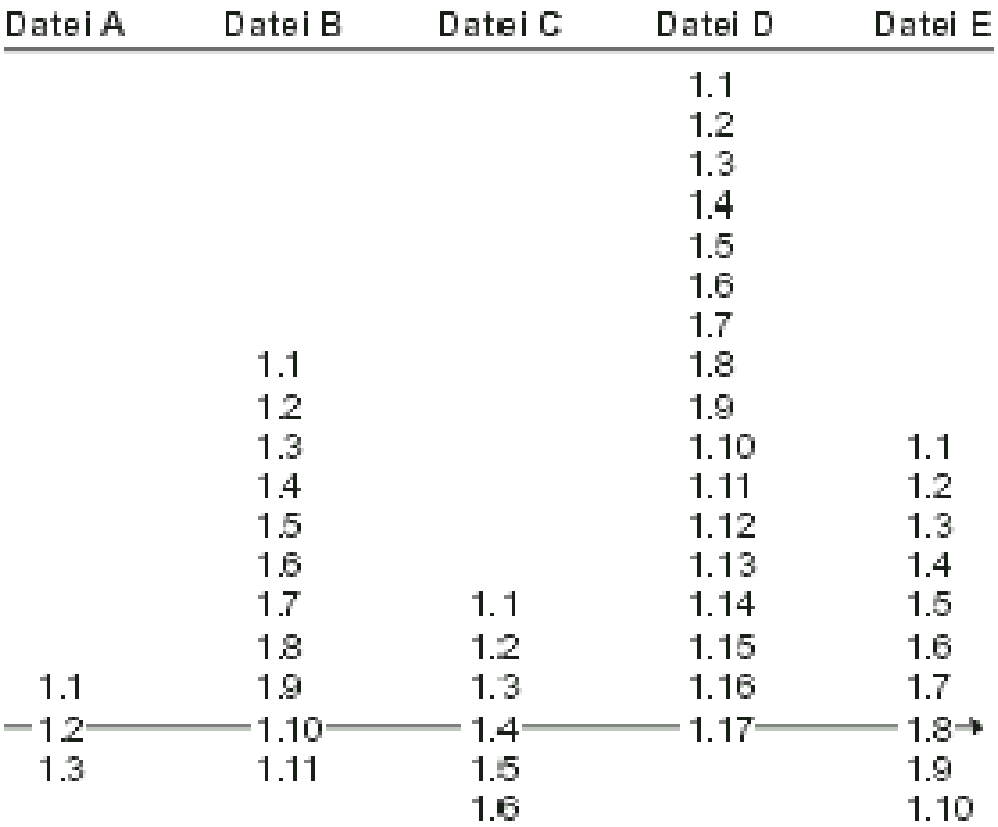
Hinweis

- Der Befehl tag bildet eine Gruppe von Revisionen, repräsentiert durch die aktuelle Arbeitskopie, auf einen Tag-String ab
- Über Tags kann auf die Gruppe von Revisionen zugegriffen werden
- Tags sind mit einer zeitlichen Momentaufnahme vergleichbar
- Kein commit beim Taggen notwendig

Einen Zeitpunkt markieren



Einen Zeitpunkt markieren



Benutzung der Momentaufnahmen

Beispiele für Zugriff

```
> cvs checkout -r rel-01 projekt01
> cvs diff -r rel-01 main.c
> cvs update -r rel-01
```

Sticky Tags

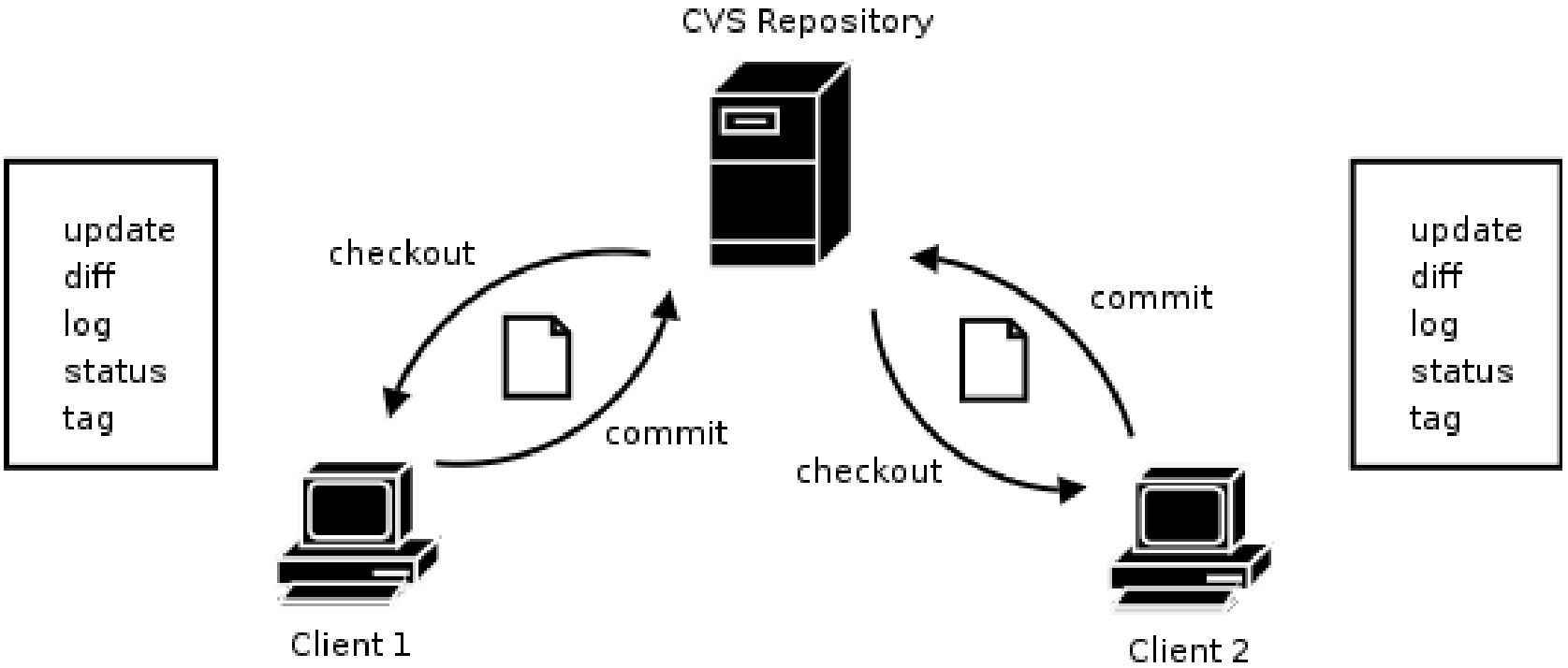
Vorgehensweise

```
> cvs checkout -r rel-01
> vi projekt01/main.c
> cvs commit
cvs commit: Examining .
cvs commit: sticky tag 'rel-01' for file ...
cvs [commit aborted]: correct above errors first!
```

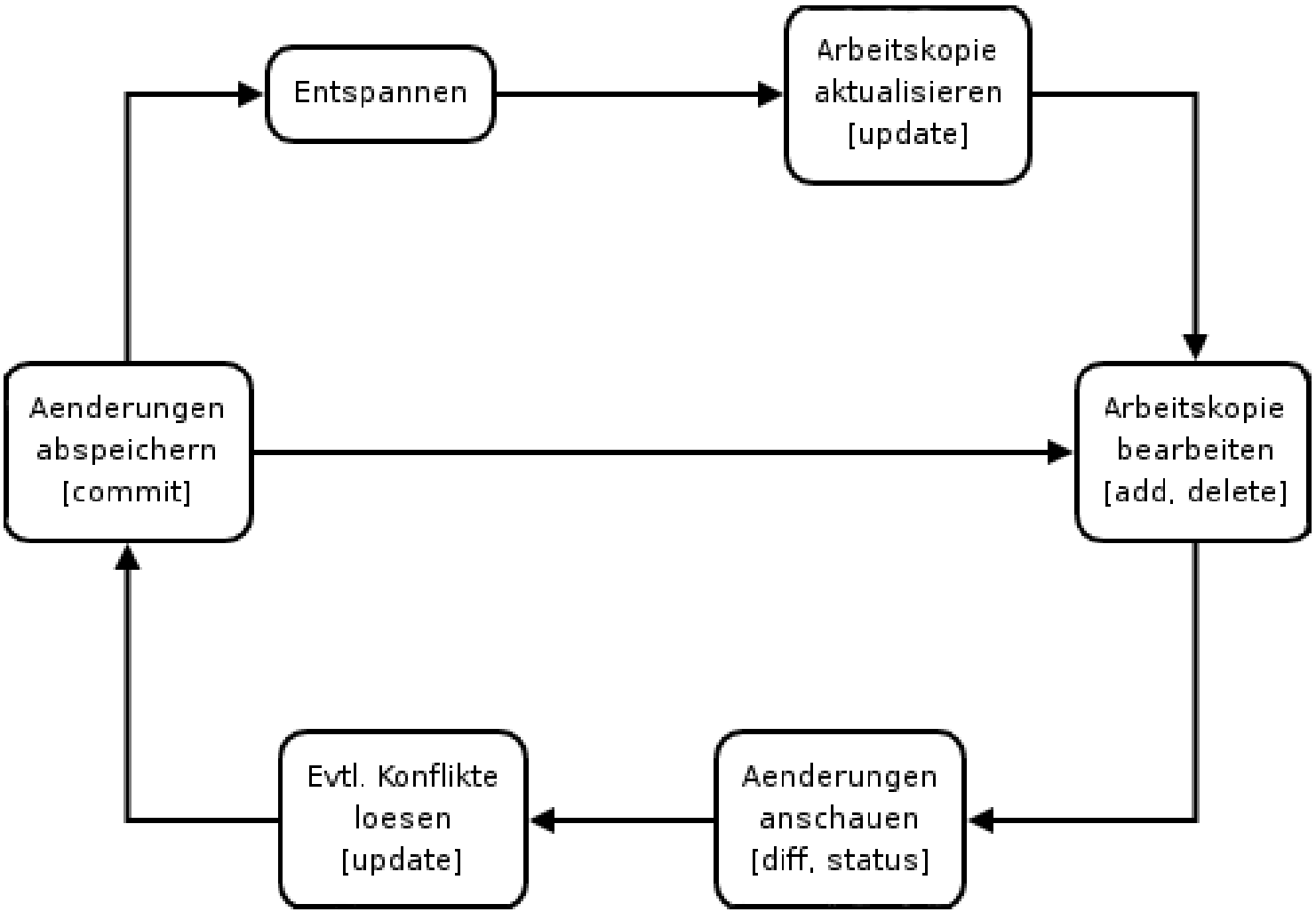
Hinweis

- Die Vergangenheit kann man nicht verändern!
- Ausweg:
  - cvs update -A
  - Verzweigungen

Zusammenfassung



CVS Befehlszyklus



Nachteile von CVS

Hinweis

- Verschieben und Umbenennen von Verzeichnissen/Dateien ist umständlich und bringt unerwünschte Nebenwirkungen mit sich
- Keine Verwaltung von Verzeichnissen und Metadateien wie Dateiberechtigungen
- Umständlicher Umgang mit Binärdateien
- Commits sind nicht atomar und können somit zu inkonsistenten Zuständen führen
- Operationen wie 'update', 'tagging' und 'branching' sind recht langsam
- Windows und Linux Clients behindern sich gegenseitig

Links und Quellen

Wo gibt es CVS?

- http://www.nongnu.org/cvs/ - Offizielle Homepage
- CVS ist Bestandteil jeder Linux Distribution
- http://ximbiot.com/ - Dokumentation
- http://cvsbook.red-bean.com/ - deutsche Dokumentation
- Man Pages

Platformunterstützung

- Clients für Linux und Windows verfügbar
- Server nur für Unix/Linux verfügbar

Ende

Danke für die Aufmerksamkeit!