CVS Einführung

- Vorteile eines Versions-Kontroll-Systems
- Umgang mit CVS
- Repositories
- Mehrere Nutzer und Konflikte

Birgit Möller & Denis Williams
AG Bioinformatik & Mustererkennung
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg

Vorteile eines Versions-Kontroll-Systems

- Scenario: n files in einem Verzeichnis werden über längere Zeit bearbeitet.
  Aufgabe der Versionskontrolle:
  - Files sichern
  - Änderungen tracken
  - Änderungen meherer Nutzer konsistent halten
- CVS macht das mit Hilfe einer *Sandbox*:
  jeder Nutzer hat eine eigene Kopie der Files, die er beliebig verändern kann.
- Verwaltet werden die Files von CVS in einem *Repository*.
- Jeder Nutzer kann nun Änderungen in seinen Files ins Repository einchecken.
  Das File erhält dabei jeweils eine neue Versionsnummer (*Revision*).

Vorteile eines Versions-Kontroll-Systems

- Man kann jederzeit:
  - jede Revision eines Files zurückbekommen (auschecken)
  - Änderungen zwischen beliebigen Revisionen (und zum aktuellen File) ansehen
  - Herausfinden welcher Nutzer welche Änderung gemacht hat (und Kommentare zu den Änderungen ansehen)

Umgang mit CVS

- Sandbox aus Repository auschecken:

```
cvs -d cvsroot co repDir Prj/ |– CVS ‘– bspProjekt |– CVS ... |– file1 ‘– unterverzeichnis |– CVS ... ‘– file2

>cvs -d /tmp/bspcvsroot co Prj/bspProjekt
cvs checkout: Updating Prj/bspProjekt
U Prj/bspProjekt/file1
cvs checkout: Updating Prj/bspProjekt/unterverzeichnis
U Prj/bspProjekt/unterverzeichnis/file2
```

```
Prj/
|-- CVS
‘-- bspProjekt
    |-- CVS ...
    |-- file1
    ‘-- unterverzeichnis
        |-- CVS ...
        ‘-- file2
```

- Ein neues file in der Sandbox zum Repository zu Verwaltung hinzufügen:

```
cvs add filename

>cvs add file2
cvs add: scheduling file ‘file2’ for addition
cvs add: use ’cvs commit’ to add this file permanently
```

Umgang mit CVS

- Änderungen übernehmen:

```
cvs commit -m "Kommentar" filename

>cvs commit -m "mein 2. tolles file"                | file2 bei commit:
cvs commit: Examining .                             | Test File2
cvs commit: Examining unterverzeichnis              |
RCS file: /tmp/bspcvsroot/Prj/bspProjekt/file2,v    | danach ändern zu:
done                                                | Dies ist
Checking in file2;                                  | Test File2
/tmp/bspcvsroot/Prj/bspProjekt/file2,v  <--  file2  |
initial revision: 1.1                               |
done
```

- Änderungen zwischen aktueller Version im Repository und der Sandbox anzeigen:

```
cvs diff filename

>cvs diff file2
Index: file2
===================================================================
RCS file: /tmp/bspcvsroot/Prj/bspProjekt/file2,v
retrieving revision 1.1
diff -r1.1 file2
0a1
> Dies ist
```

Umgang mit CVS

- Geänderte Files im Repository anzeigen (-n = nichts verändern):

```
cvs -n update

>cvs -n update                                | file1 von jemand anderem geändert von
cvs update: Updating .                        |   Das hier ist flasch geschrieben!
U file1                                       |
M file2                                       | zu
cvs update: Updating unterverzeichnis         |   Das hier ist richtig geschrieben!
```

- Änderungen im Repository in die Sandbox übernehmen:

```
cvs update filename

>cvs update file1                             | file1 danach:
U file1                                       | Das hier ist richtig geschrieben!
```

Repositories

- Repository:
  - CVS speichert hier die Files.
  - Beliebig viele anlegbar.
  - Wird normalerweise nie direkt bearbeitet.
  - Enthält *CVSROOT* Verzeichnis mit Administrativen Daten.

- Jedes Verzeichnis einer Sandbox
  - enthält ein Verzeichnis CVS mit Verwaltungsdateien
  - gehört zu genau einem Repository (steht in CVS/Root)

- Anlegen von Repositories:

```
cvs -d cvsroot init
```
Legt in cvsroot Verzeichnis CVSROOT an mit nötigen administrativen files.

```
cvs -d /tmp/bspcvsroot init
```

Umgang mit CVS

- Import einer Dateistruktur:

```
cvs -d cvsroot import -m"Kommentar" repDir vendor-tag release-tag

>cvs -d /tmp/bspcvsroot import -m "Import des Beispiels" Prj/bspProjekt williams initial
N Prj/bspProjekt/file1
cvs import: Importing /tmp/bspcvsroot/Prj/bspProjekt/unterverzeichnis
N Prj/bspProjekt/unterverzeichnis/file2

No conflicts created by this import
```

Mehrere Nutzer und Konflikte

- Konflikte entstehen, wenn mehrere Nutzer an den selben Stellen ändern
- Nur der erste kann seine Daten einchecken (mit *commit*).
- Wurde ein file im Repository geändert, muss erst die Sanbox auf den neuesten Stand gebracht werden (mit *update*)

```
file3:                    | "Änderung von Nutzer1 eingecheckt:  |  Änderung von Nutzer2:
1 Hier wird               | 1 Hier wird                         |  1 Hier wird
2 ein Konflikt            | 2 ein von Nutzer1 Konflikt          |  2 ein Konflikt
3 produziert              | 3 produziert                        |  3 produziert
                          | 4 von Nutzer2                       |
Nutzer2>cvs update
cvs update: Updating .
RCS file: /tmp/bspcvsroot/Prj/bspProjekt/file3,v
retrieving revision 1.1
retrieving revision 1.2
Merging differences between 1.1 and 1.2 into file3
M file3
cvs update: Updating unterverzeichnis
                                                            |  file3 von Nutzer2 danach:
                                                            |  1 Hier wird
                                                            |  2 ein von Nutzer1 Konflikt
                                                            |  3 produziert
                                                            |  4 von Nutzer2
```

Mehrere Nutzer und Konflikte

```
Neue Änderung von Nutzer1            |  Änderung von Nutzer2 eingecheckt:
1 Hier wird                          |  1 Hier wird
2 ein von Nutzer1 Konflikt           |  2 ein von Nutzer1 Konflikt
3 produziert von Nutzer1             |  3 produziert von Nutzer2
4 von Nutzer2                        |  4 von Nutzer2

Nutzer1>cvs update
cvs update: Updating .
RCS file: /tmp/bspcvsroot/Prj/bspProjekt/file3,v
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into file3
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in file3
C file3
cvs update: Updating unterverzeichnis

file3 von Nutzer1 danach: KONFLIKT!
1 Hier wird
2 ein von Nutzer1 Konflikt
<<<<<<< file3
3 produziert von Nutzer1
=======
3 produziert von Nutzer2
4 von Nutzer2
>>>>>>> 1.3

Jetzt muss Nutzer1 den Konflikt von Hand lösen und kann dann das file3 einchecken.
```