8. Dokumentenverwaltung mit CVS – eine Einführung

8.1 Warum CVS?

8.2 Basiskonzepte

8.3 Vorbereitungen

8.4 Anlegen eines neuen Projekts

8.5 Bearbeiten eines Projekts

8.1 Warum CVS?

CVS = Concurrent Versions System

Problemstellung

Innerhalb einer Projektgruppe wird an verschiedenen Dokumenten (Quelltexte, Dokumentation) gemeinsam gearbeitet. Es muß erkennbar sein, wer welche Änderung/Ergänzung vorgenommen hat. Ggf. ist es notwendig Änderungen wieder rückgängig zu machen, d.h. zur einer alten Version zurückzukehren.

Möglichkeiten zur Dokumentenverwaltung

1. **Verwaltung in privaten Verzeichnissen**
   Jeder verwaltet ein eigenes Verzeichnis mit den entsprechenden Dokumenten. Geänderte Dokumente werden per e-mail o.ä. ausgetauscht. Probleme: Wer darf was modifizieren? Wer hat die aktuelle Version?

2. **Verwaltung in einem gemeinsamen Verzeichnis**
   Alle teilen sich ein gemeinsames Verzeichnis. Änderungen werden sofort bei jedem sichtbar. Probleme: Gegenseitiges Überschreiben von Dateien.

8.2 Basiskonzepte

Dokumenttypen

Mit CVS können beliebige Dateien verwaltet werden, wobei der Schwerpunkt auf Textdateien liegt. In der Regel werden nur Quelldateien (*.java, *.c, *.tex) verwaltet – nicht die daraus abgeleiteten (*.class, ...).

Archivierung in zentralem Repository

Für CVS ist ein Projekt einfach ein Dateibaum. An zentraler Stelle, dem sog. Repository, werden alle Fassungen der einzelnen Dateien zusammen mit Änderungsprotokollen archiviert.

Arbeiten auf lokalen Kopien

Einzelne Benutzer besitzen jeweils lokale Kopien, die sie wie gewohnt bearbeiten können, insbesondere unabhängig von anderen Entwicklern.

Zusammenführen von Änderungen

Lokale Änderungen werden vom Benutzer in das Repository übertragen. CVS sorgt dafür, dass Änderungen verschiedener Benutzer an derselben Dateien zusammengeführt werden.

8.3 Vorbereitungen (1)

Ort des Repository

Allgemein:

```
:accessmethod:user@server:path
```

Im Programmierpraktikum:

accessmethod = pserver

user = (durch CVS-Login ersetzen)

server= eratosthenes.informatik.uni-mannheim.de

path = /opt/cvs/cvsroot/ProgrammierMethodik/pmXX

(XX durch die Gruppennummer ersetzen)

Kommandozeile (Setzen von CVSROOT):

Unix:

```
export CVSROOT=:accessmethod:user@server:path
```

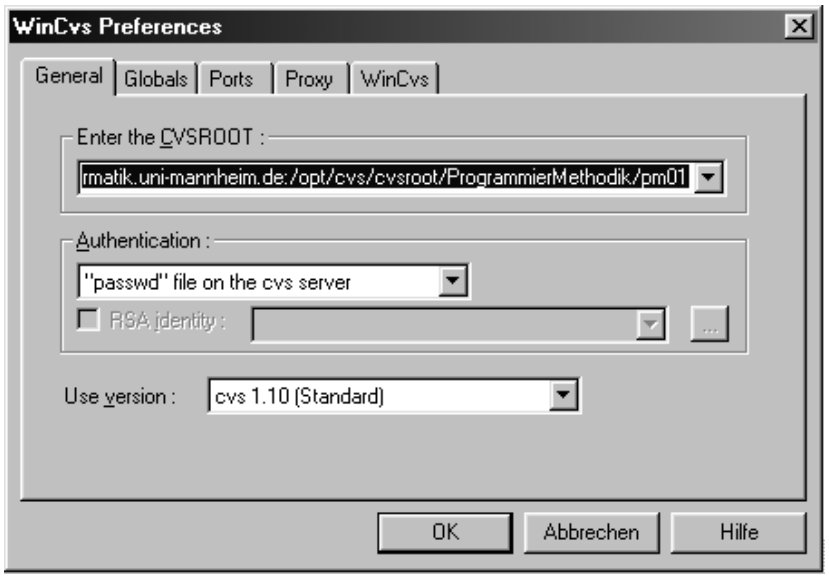
Windows:

```
set CVSROOT=:accessmethod:user@server:path
```

8.3 Vorbereitungen (2)

Unter WinCVS:

Menüpunkt „Admin – Preferences“



1. „General – Enter the CVSROOT“:
   Repository angeben (ohne :pserver:)
2. „General – Authentication“: „passwd“ file on the cvs server
3. „Globals – Checkout read only“: deaktivieren

8.4 Anlegen eines neuen Projekts (1)

Projekt lokal anlegen:

Verzeichnisstruktur lokal anlegen, z.B.

```
cluedo
cluedo/src
cluedo/doc
```

Pro Verzeichnis eine „Dummy“-Datei anlegen:

```
cluedo/src/CluedoServer.java
cluedo/doc/README.TXT
```

Projekt in das Repository importieren:
Kommandozeile:

```
cd cluedo
cvs import -m "comment" cluedo pmXX start
```

Projektname

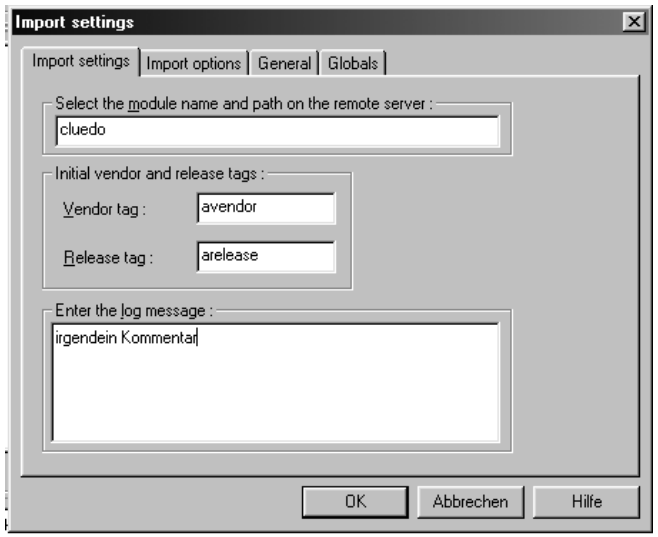
vendor tag

release tag

8.4 Anlegen eines neuen Projekts (2)

Unter WinCVS:

1. Links in der Verzeichnistruktur das Verzeichnis markieren.
2. Menüpunkt „Create – Import module“
3. Im Dialogfenster „Ordner suchen“ erneut das Verzeichnis markieren
4. Im Dialogfenster „Import filter“ continue anwählen.



1. „Vendor tag“: beliebig (z.B. pmXX)
2. „Release tag: z.B. start

8.5 Bearbeiten eines Projekts (1)

CVS-Kommandos (Auszug):

1. **checkout**
   Erstellen einer lokalen Kopie, die dann bearbeitet werden kann
2. **commit**
   Änderungen an Dateien in das Repository eintragen
3. **add**
   Neue Dateien zum Repository hinzufügen
4. **remove**
   Dateien aus dem Repository löschen
5. **update**
   Änderungen anderer Benutzer in die lokale Kopie übernehmen.

Erstellen einer lokalen Kopie (1)

Lokale Kopie erstellen („checkout“):

Damit wird eine Kopie der aktuellen Version im Repository in einem lokalen Verzeichnis erzeugt.

Kommandozeile:

```
cd mysrcdir
cvs checkout cluedo
```

Projektname/
Modulname

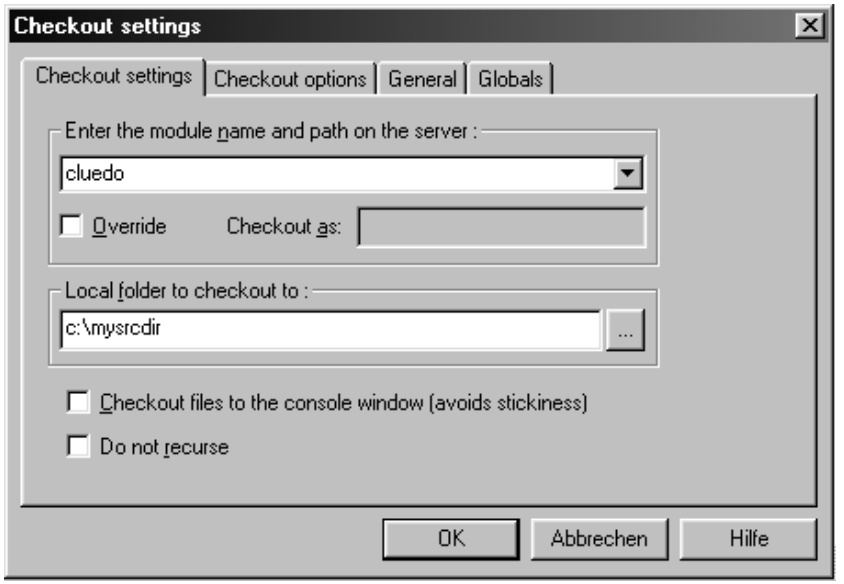
Das erzeugt im Verzeichnis „mysrcdir“, eine Kopie der aktuellen Version:

```
cluedo
cluedo/src
cluedo/doc
cluedo/src/CluedoServer.java
cluedo/doc/README.TXT
```


Erstellen einer lokalen Kopie (2)

Lokale Kopie unter WinCVS:

1. „Create – Checkout module ...“
2. Unter „Enter the module name ...“ den Namen des auszucheckenden Moduls angeben
3. Unter „Local folder ...“ angeben wo die Dateien hinkopiert werden sollen.



Ändern von Dateien innerhalb des Projekts (1)

Checkout:

Nach dem „Checkout“ ist eine komplette Kopie lokal verfügbar.

Edit:

Die Dateien können nun mit einem beliebigen Editor bearbeitet werden.

Commit:

Die veränderte Version muss dann noch an das zentrale Repository übergeben werden („commit“).

Wir nehmen an, dass wir die Datei CluedoServer.java verändert haben.

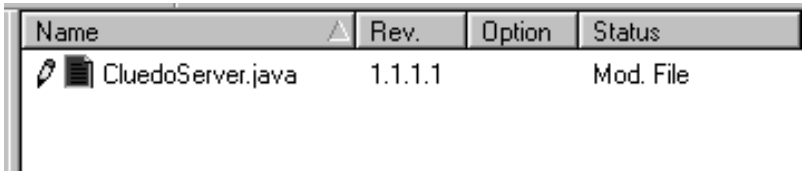
Kommandozeile:

```
cvs commit -m "comment" CluedoServer.java
```


Ändern von Dateien innerhalb des Projekts (2)

„Commit“ unter WinCVS:

1. Veränderungen an Dateien werden durch das rote Blatt-Icon am Zeilenanfang symbolisiert. Desweiteren steht unter Status „modified file“.
2. Rechte Maustaste auf der markierten Datei und „Commit selection ...“ auswählen.
3. Kommentartext unter „log message“ eingeben



Hinzufügen von Dateien innerhalb des Projekts (1)

Datei erstellen:

Mit einem beliebigen Editor eine neue Datei in einem Projektverzeichnis erstellen.

Add:

Dem Repository muss mitgeteilt werden, dass eine Datei hinzugekommen ist („add“). Danach muss diese Version an das zentrale Repository übergeben werden („commit“).

Wir nehmen an, dass wir die Datei CluedoClient.java neu erstellt haben.

Kommandozeile:

```
cvs add –m "comment" CluedoClient.java
cvs commit –m "comment" CluedoClient.java
```

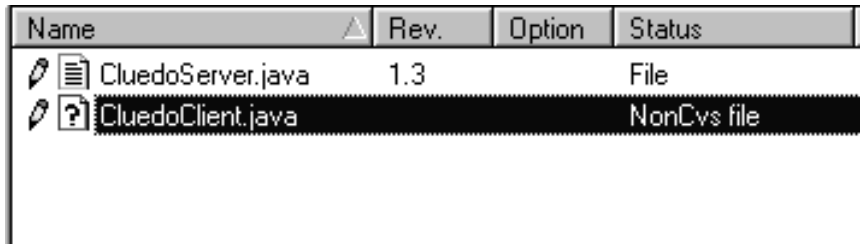
ACHTUNG:

Beim Hinzufügen von Binärdateien muß zusätzlich die Option –kb (vor –m) angegeben werden.

Hinzufügen von Dateien innerhalb des Projekts (2)

„Add“ unter WinCVS:

1. Eine neue Datei wird duch ein Icon mit einem Fragezeichen am Zeilenanfang symbolisiert. Desweiteren steht unter Status „NonCVS file“.
2. Neue Datei markieren und aus der Button-Leiste den Knopf „Add selected“ auswählen. Bei einer Binärdatei ist entsprechend „Add select binary“ anzuklicken.
3. Danach ist die Datei als „modified“ gekennzeichnet. Sie muss mit „commit“ noch in das Repository eingetragen werden.



Löschen von Dateien innerhalb des Projekts (1)

Remove:

Dem Repository muss mitgeteilt werden, dass eine Datei gelöscht werden soll („remove“). Auch dieses muss danach wieder mit dem Befehl „commit“ bestätigt werden.

Wir nehmen an, dass wir die Datei CluedoClient.java löschen wollen.

Kommandozeile:

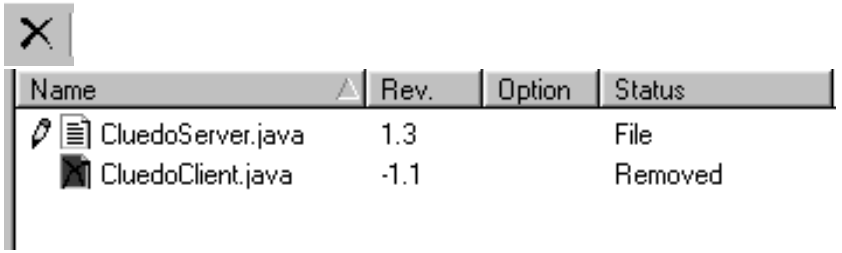
```
del CluedoClient.java
cvs remove CluedoClient.java
cvs commit -m "comment" CluedoClient.java
```


Löschen von Dateien innerhalb des Projekts (2)

„Remove“ unter WinCVS:

1. Zu löschende Datei markieren und aus der Button-Leiste den Knopf „Remove selected“ auswählen.

2. Danach ist die Datei als „removed“ gekennzeichnet. Sie muss mit „commit“ noch in das Repository eingetragen werden.



Änderungen anderer Benutzer innerhalb des Projekts übernehmen (1)

Update:

Vom Repository werden die Veränderungen im Vergleich zur eigenen lokalen Version angefordert.

Dieses Update ist auch für ganze Verzeichnisse möglich.

Wir nehmen an das wir das gesamte Verzeichnis src auf neuesten Stand bringen wollen.

Kommandozeile:

```
cd cluedo\src
cvs update
```


Änderungen anderer Benutzer innerhalb des Projekts übernehmen (2)

„Update“ unter WinCVS:

1. Verzeichnis oder Dateien markieren, die auf neuesten Stand gebracht werden sollen.

2. Rechte Maustaste auf der Selektion und „Update selection“ wählen.

Beheben von Konflikten (1)

Konflikte treten dann auf, wenn zwei Benutzer dieselbe Datei editieren.

In der Regel kann CVS die Veränderungen in eine gemeinsame Version integrieren.

Überschneiden sich die Änderungen jedoch, muss der Benutzer eingreifen.

Beispiel:

Version im Repository / CluedoServer.java:

```
public class CluedoServer{
}
```

Benutzer 1 / CluedoServer.java:

```
class CluedoServer {
}
```

Benutzer 2 / CluedoServer.java:

```
static class CluedoServer {
}
```


Beheben von Konflikten (2)

Ablauf:

1. Benutzer 1 checkt seine Version ein:
   `cvs commit -m "" CluedoServer.java`
2. Benutzer 2 will seine Version einchecken:
   `cvs commit -m "" CluedoServer.java`
   Er erhält folgende Fehlermeldung (auch bei WinCVS):
   `cvs-server: Up-to-date check failed for ...`
   Das bedeutet, dass zwischenzeitlich im Repository Änderungen erfolgt sind.
3. Benutzer 2 aktualisiert seine Version:
   `cvs update CluedoServer.java`
   Er erhält folgende Warnung:
   `warning: conflict during merge`
4. Benutzer 2 behebt den Konflikt und checkt die neue Version ein.

Beheben von Konflikten (3)

Datei CluedoServer.java mit Konflikt:

```
<<<<<<< CluedoServer.java
static class CluedoServer {
=======
class CluedoServer {
 >>>>>>> 1.8
}
```

Der Konflikt ist mit „<<<<<“ und „>>>>>“ gekennzeichnet.