Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Prof. Dr. W.-P. de Roever
Karsten Stahl, Martin Steffen

Programming-in-the-many (Java)

Sommersemester 2002 — **Handout 2** — 5. April 2002

Handout 2: Koordination mit CVS

Ausgabetermin: 5. April 2002

Verteiltes Arbeiten unter CVS

Zu Koordination der einzelnen Gruppen ist die Verwendung von "*Concurrent Version Control*" (CVS) vorgeschrieben. Wir werden eine kurze Einführung in CVS geben und einen *account* als Server für das *Repositorium* zur Verfügung stellen. Die wichtigesten Dinge sind auf diesem Handout zusammengefaßt. Weitere Informationen über unsere Webseite.

Technische Vorraussetzungen

Man benötigt zwei Dinge, um von Ferne auf das gemeinsame Repositorium zugreifen zu können:

1. *ssh* (secure shell) für den sicheren Zugriff und
2. *cvs* + *rcs* als Versionskontrolle.

Diese Tools sind auf dem Uni-Netz vorhanden. Wer daheim (oder von einem sonstigen account) arbeiten will, muß dafür sorgen, daß sie installiert sind. Sie sind kostenlos für Linux/Unix/Windows/MacOS verfügbar.

ssh-Zugriff

Für den ersten Punkt, dem Fernzugriff unter *ssh*, geht man wie folgt vor. Man beschaffe sich einen *ssh*-Schlüssel Der Schlüssel liegt typischerweise unter `~/.ssh/identity.pub`. Diesen maile man an `swprakt@informatik.uni-kiel.de`. Wer einen solchen Schlüssel nicht hat, muß ihn sich generieren und zwar mittels *ssh-keygen* am besten `ssh-keygen -tdsa` für einen ssh-2 Schlüssel.

Angabe des cvs-Servers

Für das korrekte Funktionieren muß man cvs noch mitteilen, wo sich das Repositorium befindet. Dies kann man wie folgt erreichen (bash-Syntax):

```
export CVSROOT=swprakt@goofy.informatik.uni-kiel.de:/home/swprakt/cvsroot
export CVS_RSH=ssh
export CVSEDITOR=emacs

export CLASSPATH=$WORKDIR/Slime/src:
```

wobei `$WORKDIR` ein Platzhalter (!) für das Verzeichnis ist, in dem man arbeiten wird. Die ersten drei Zeilen sind zum korrekten Steuern von cvs gedacht.[1] Anstelle *emacs* kann man auch *emacsclient* oder den Editor seiner Wahl nehmen. *emacsclient* sollte man dann nehmen, wenn man ohnehin emacs verwendet und beim starten des emacs' ein `(server-start)` ausgeführt wird. Die letzte Zeile hat nichts mit ssh/cvs, sondern hat mit dem diesjährigen Java-Projekt selbst zu tun.

Beispiel

Das Beispiel verwendet als Bezeichnung für das *Arbeitsverzeichnis* Als `$WORKDIR` kann sich jeder selbst passend wählen, wo er das Arbeitsverzeichnis haben will. Es spricht nichts dageben, daß man sich auch mehrere Arbeitsverzeichnisse verschafft, zum Beispiel eines auf seinem Uni-account und eines daheim, oder daß ein Team zwei Arbeitsverzeichnisse hat, an denen es getrennt arbeitet. Mit der Zahl der ausgecheckten Arbeitsverzeichnisse steigt natürlich die Möglichkeiten der Verwirrung. In der Regel kommt man mit einem Arbeitsverzeichnis pro Person und pro "Arbeitsplatz" aus.

- **Auschecken:** So bekommt man zum ersten Mal seine Arbeitskopie des SLIME-Projektes:

  ```
  mkdir $WORKDIR
  cd $WORKDIR
  cvs checkout Slime
  ```

  Die Variable `$WORKDIR` ist hier nur zur Illustration gewählt, cvs kennt sie nicht. Falls man konkret `$WORKDIR = ~/Projekt` wählt, heißen die Befehle:

  ```
  cd ~
  mkdir Projekt
  cd Projekt
  cvs checkout Slime/src
  ```

  Im Folgenden bezeichnet `$WORKDIR` immer das frei wählbare Arbeitsverzeichnis. Macht man

  ```
  cvs checkout Slime/org
  ```

[1] Man kann cvs auch anders steuern, z.B. über .cvsrc, bei Bedarf bitte selbst im Manual nachschlagen oder nachfragen.

bekommt man die Gruppeneinteilung. Macht man nur

```
cvs checkout Slime
```

so bekommt man alles zum SLIME-Projekt, einschließlich der Quellen des Pflichtenheftes und der Web-Seite.

- **Zurückspeichern:** Wenn man mit den Änderungen durch ist, kann man den gesamten Verzeichnisbaum zurückspeichern:[2]

```
cd $WORKDIR/Slime
cvs commit
```

Danach wird man aufgefordert, einen Kommentar bezüglich seiner Änderungen anzugeben. Falls der Kommentar nur kurz ist, kann man schneller auch

```
cvs commit -m"Anmerkungen zu den Änderungen"
```

verwenden.

- **Updaten:** Den Baum seines Arbeitsverzeichnisses auf den neuesten Stand bringen, geht so

```
cd $WORKDIR/Slime
cvs update
```

- **Neue Dateien + Verzeichnisse:** Das geht mit

```
cvs add [filename]
```

entfernen mit

```
cvs remove [filename],
```

Anstelle eines Dateinamens kann man auch ein Verzeichnis hinzufügen oder auch alle neuen java-Dateien mit `cvs add *.java`.

Strategie und Spielregeln

Versionskontrolle garantiert kein reibungsloses Arbeiten, es unterstützt dies, wenn man gewisse Disziplin wahrt. Folgende Daumenregeln:

- Im Allgemeinen gilt: sobald die Gefahr besteht, daß eine Änderung die anderen Gruppen in Mitleidenschaft ziehen kann, soll dies in der Regel vorher abgeklärt werden. Insbesondere:
  - Ändern von globalen Paketen/Paketen von anderen Gruppen: nur nach reiflicher Überlegung und Rücksprache mit der betroffenen Gruppe.

[2] Beachte den Wechsel in das Unterverzeichnis.

  - Keine unangekündigte Änderung der *Verzeichnisstruktur* im Repositorium. Neue Unterverzeichnisse im eigenen Paket sind dabei in Ordnung. Man soll auch die Finger von den administrativen cvs-Dateien lassen.
  - kein globales Rückgängigmachen von Änderungen anderer Seite ohne Rücksprache.
  - Keine "*watches*" setzen (außer eventuell auf seinen eigenene Code).

- Nur *kompilierbare* Versionen einchecken, d.h. der eigene Teil muß sich mittels `make all` ohne Fehlermeldungen kompilieren lassen. Dies gilt noch nicht für die Anlaufphase, bis die Pakete zum ersten Mal integriert werden. Änderungen, die die *Schnittstellen* mit einem anderen Paket betreffen sollten *angekündigt* werden, zum Beispiel in den Besprechungen. Daneben sind die Mailadressen der einzelnen Gruppen und ihrer Teilnehmer im Netz vorhanden.

- `Makefile`s und `Readme`s sind hilfreich. Der Code für jedes Pakets soll mittels `make all` als erstes "target" des Makefiles kompilierbar sein. Im Paket `absynt` finden sich Beispiele.

- Der Code soll sinnvoll mittels *Javadoc* kommentieren werden. Zumindest der Author/die Autoren sollen dokumentiert sein. In Paket `absynt` kann man sich anschauen, wie man auch cvs-logs dokumentieren lassen kann, (z.B. indem man die Datei `templates/classtemplate.txt` verwendet und anpaßt.) Die generierte Dokumentation wird in regelmäßigen Abständen im Netz bereitgestellt.

Literatur

[CVS01a] Concurrent versions systems: The open standard for version control. available at `http://www.cvshome.org/`, 2001.

[CVS01b] *Concurrent Versions Systems: CVS Manual*, 2001. available at `http://www.cvshome.org/docs/manual/cvs.html`.

[Fog00] Karl Fogel. *Open Source Projekte mit CVS*. MITP-Verlag, 2000.