Universität Koblenz-Landau WS '04/05
Institut für Informatik
Bernhard Beckert · www.uni-koblenz.de/~beckert
Vladimir Klebanov · www.uni-koblenz.de/~vladimir

Praktikum

Entwicklung objektorientierter Software mit formalen Methoden

Aufgabenblatt 1

Aufgabe 1

Machen Sie sich mit dem Revisionskontrollsystem CVS vertraut. Dieses bietet neben vielen weiteren Vorteilen, wie dem Rücksetzen auf einen älteren Stand oder dem Verfolgen von Änderungen, unter anderem die Möglichkeit, parallel an einer Aufgabe zu arbeiten. Eine sehr empfehlenswerte (und frei verfügbare) Anleitung für CVS ist "Open Source Development With CVS" von Karl Fogel, zu finden unter: `http://cvsbook.red-bean.com/`

Das CVS-Repository für das Praktikum befindet sich unter `/home/vladimir/PRAKTCVS`. Falls Sie lokal arbeiten, könen Sie diesen Pfad CVS direkt über den Parameter `-d` oder die Umgebungsvariable `CVSROOT` mitteilen.

Als erstes führen Sie folgende Kommandos aus:

```
mkdir praktikum
cd praktikum
mkdir -p uebungsblaetter/1/
```

Stellen Sie nun sicher, dass Sie sich im Unterverzeichnis `praktikum` befinden und führen Sie folgenden Befehl aus:

`cvs import -m "Initialer Check-In" gruppe<nr> keyprkt<nr> start`

import veranlasst CVS, alle Verzeichnisse und Dateien, die es unterhalb der aktuellen Position findet, in das Repository aufzunehmen.

-m spezifiziert einen Kommentar, der kurz die Änderungen beschreibt.

gruppe<*nr*> wird das neue Top-Level Verzeichnis

keyprkt<*nr*> ist die Urheber-Identifizierung

start ist eine initiale Markierung ("Tag"), unter der sich der soeben abgelegte Stand wieder zurückholen läßt.

Verlassen Sie das Verzeichnis `praktikum` und legen Sie zwei neue Verzeichnisse, zum Beispiel `mueller` und `meier`, an. Wechseln Sie nach `mueller` und führen Sie folgenden Befehl aus:

`cvs co gruppe<nr>`

ein anschließendes `ls gruppe<nr>/` sollte folgendes Bild liefern:

```
CVS uebungsblaetter
```

Wechseln Sie in das Unterverzeichnis `praktikum` und legen Sie dort eine Textdatei `Gruppe<nr>.txt` an, in der Sie die Namen und Emailadressen der Mitglieder Ihrer Gruppe ablegen.

Als nächstes Markieren Sie sie die hinzuzufügende Datei mit dem Befehl:

```
cvs add Gruppe<nr>.txt
```

Beachten Sie bitte, dass die bisherigen Änderungen, noch nicht vom Repository übernommen wurden, dafür ist ein zusätzlicher Arbeitsschritt, das sogenannte Einchecken, notwendig.

Führen Sie vor jedem Einchecken zunächst eine Aktualisierung `cvs update -d` Ihres Arbeitsverzeichnisses (also z.B. `mueller`) durch. Somit ist sichergestellt, dass vor dem Einchecken die von anderen Mitgliedern der Gruppe veränderten Dateien berücksichtigt werden. Ein abschließendes

```
cvs commit -m "dies und jenes wurde geaendert"
```

legt die Änderungen im Repository mit dem übergebenen Kommentar (`-m`) ab.

Wechseln Sie ins Verzeichnis `meier`. Sofern Sie im vorherigen Arbeitschritt wie oben noch keine Version ausgecheckt haben, machen Sie das jetzt, andernfalls reicht auch ein einfaches `cvs update`. Anschließend sollte sie auch hier in dem entsprechenden Unterverzeichnis die Datei `Gruppe<nr>.txt` vorfinden. Spielen Sie nun ein bisschen mit CVS herum, indem Sie z.B. in `meier` und `mueller` die Datei `Gruppe<nr>.txt` an der gleichen (und/oder verschiedenen) Stellen abändern und Sie sie dann nacheinander einchecken.

CVS erlaubt es, Markierungen ("Tags") zu setzen, unter denen sich bestimmte Versionen, abrufen lassen. Dafür stellen Sie zunächst sicher, dass alle Dateien in Ihrem Arbeitsverzeichnis in der vorliegenden Version, bereits im Repository vorhanden sind (also schon einmal eingecheckt wurden). Eine Markierung setzen Sie mit:

`cvs tag` <name>

Damit lässt sich dann über `cvs co -r` <*name*> `praktikum` genau diese Version wieder auschecken.

Aufgabe 2

Beschreiben Sie ein System, das die Konferenanmeldung folgendermaßen unterstützt, möglichst genau mit UML in TogetherCC.

Bei einer Konferenz muß zu jedem Papier ein Autor einen Vortrag halten. Jeder Teilnehmer muß sich zuerst registrieren lassen und erhält eine Rechnung für den Konferenzbeitrag, sowie eine Quittung nach erfolgter Zahlung. Außerdem kann man sich zu Workshops, Tutorials und Social Events anmelden. Die Konferenzveranstalter können bestimmte Veranstaltungen absagen (z.B. bei Teilnehmermangel) und die Gebühren erstatten.

(a) Erstellen Sie ein UML-Use-Case-Diagramm

(b) Erstellen Sie ein UML-Klassendiagramm.

(c) Erstellen Sie ein UML-Sequenzendiagramm.

(d) Erstellen Sie ein UML-Zustandsmaschinen-Diagramm, das den Zustand einer Anmeldung beschreibt.

Aufgabe 3

In dem Buch *A Programming Approach to Computability* von Kfoury, Moll und Arbib (Springer Verlag 1982) findet sich auf S. 19f die folgende Definition einer einfachen (künstlichen) Programmiersprache:

> We define a **while**-program to be a finite (possibly empty) sequence of *statements*, separated by semicolons, and preceded by **begin** and followed by **end**:
>
> $$\textbf{begin } S_1; S_2; \ldots; S_n \textbf{ end}$$
>
> where $S_1, S_2, \ldots, S_n$ are arbitrary statements and $n \geq 0$.
>
> Statements are defined inductively. The "assignment statements" form the basis step, while the induction step involves the formation of "**while** statements" and "compound statements".
>
> In **while**-programs, the values assigned to variables will be non-negative integers. An *assignment statement* has one of the following forms:
>
> 1. $X := 0$
> 2. $X := succ(Y)$
> 3. $X := pred(Y)$
>
> where *succ* is the "successor" function and *pred* is the "predecessor" function. [...]
>
> X and Y in the above instructions stand for arbitrary variable names, possibly identical. [...]
>
> A **while** statement has the following form:
>
> $$\textbf{while } X \neq Y \textbf{ do } S \quad ,$$
>
> which means "while the value of variable X is not equal to the value of variable Y, repeat the action specified by statement S. [...]
>
> Finally, a "compound statement" has the form:
>
> $$\textbf{begin } S_1; S_2; \ldots; S_m \textbf{ end}$$
>
> where $S_1, S_2, \ldots, S_n$ are arbitrary statements and $m \geq 0$. A compound statement such as the one given here simply means that we have to carry out the actions prescribed by $S_1, S_2, \ldots,$ and S_n in succession and in the given order.

Geben sie ein UML Klassendiagramm an, das die *abstrakte Syntax* der **while**-Sprache modelliert. D.h. die Schlüsselwörter **begin**, **end**, etc. und die Semikolons brauchen nicht modelliert zu werden. Auch die Semantik braucht nicht beachtet zu werden. Das Klassendiagramm soll lediglich die strukturellen Beziehungen der syntaktischen Elemente "program", "statement", "variable", etc. widerspiegeln. Benützen Sie das Programm TogetherCC.

Abgabe bis 16.11.

Es muss pro Gruppe nur *eine* Lösung abgegeben werden.
Die Abgabe der Übungsblätter erfolgt mit dem CVS System. Dazu legen Sie die abzugebenden Dateien im CVS ab und markieren die abzugebende Version der Dateien mit "`LoesungsBlatt`<*nr*>" wie in Aufgabe 1 beschrieben. Die Lösungen sollten vorzugsweise im dazugehörigen Unterverzeichnis `uebeungsblaetter/nr/` vorzufinden sein, zumindest aber ein Hinweis auf den Ort der Lösungen.
Einige Aufgaben verlangen eine schriftliche Bearbeitung, diese ist dann je nach Komplexität als ASCII, html, ps- oder pdf-Dokument abzugeben. Auf *keinen* Fall im MS Word `doc`-Format.

Materialien

http://www.uni-koblenz.de/~beckert/Lehre/Praktikum-Formale-Entwicklung/

Bernhard Beckert: Zi. MB 218, Tel. 287-2775, Email: beckert@uni-koblenz.de
Vladimir Klebanov: Zi. MB 224, Tel. 287-2781, Email: vladimir@uni-koblenz.de