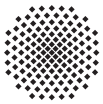

Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 3

Software Engineering (SoSe 2018)

Abgabe: Fr. 01.06.2018, 12:00 Uhr — Besprechung: Montag, 04.06.2018

- Bitte lösen Sie die Übungsaufgabe in **Gruppen von 4 Studenten** und wählen EINEN Studenten aus, welcher die Lösung im ILIAS als **PDF** (Ordner `Abgaben/Blatt 3/`) als **Gruppenabgabe** (unter Angabe aller Gruppenmitglieder) einstellt.
- Für schriftliche Aufgaben erstellen Sie EINE PDF-Datei. Die Projekte zu den Programmieraufgaben können Sie als ZIP-Datei hinzufügen. Bitte erstellen Sie ein **Titelblatt**, welches die Namen der Studenten, die Matrikelnummern und die E-Mail-Adressen enthält. Im Quellcode fügen Sie diese Informationen bitte als Kommentar hinzu.
- Benennen Sie die Dateien nach dem folgenden Schema:
  SE[**Blattnummer**]**-**[**Nachnamen der Teammitglieder**]**.**[**pdf oder zip**].

Aufgabe 1 Funktionsorientierter Test

Betrachten Sie die folgende Spezifikation einer Kaffeemaschine:

Eine Kaffeemaschine soll nur Kaffee kochen, wenn eine Kaffeekapsel eingelegt ist und eine Mindestmenge Wasser im Wasserbehälter vorhanden ist. Beides wird regelmäßig und in kurzen Abständen überprüft, falls gerade kein Kaffee gekocht wird. Die Überprüfung der Kaffeekapsel löst entweder das Ereignis *KaffeekapselVorhanden* oder *KaffeekapselFehlt* aus. Die Überprüfung des Wasserstands löst entweder das Ereignis *WasserVorhanden* oder *WasserFehlt* aus. Durch Drücken des Knopfes der Kaffeemaschine wird das Ereignis *KnopfGedrückt* ausgelöst und, falls die Vorbedingung erfüllt ist, der Kaffee gekocht. Ist der Kaffee fertig, so wird das Ereignis *KaffeeFertig* ausgelöst.

(a) Zeichnen Sie den Zustandsautomaten aus der Spezifikation der Kaffeemaschine. Dieser soll nur die beschriebenen Ereignisse berücksichtigen.

(b) Bestimmen Sie die Zustandsübergangstabelle.

(c) Leiten Sie Testfälle für eine vollstände Zustandsüberdeckung ab.

(d) Leiten Sie Testfälle für eine vollständige Zustandsübergangsüberdeckung ab.

(e) Diskutieren Sie wie, man die Tests durchführen müsste, um eine vollständige Ereignisüberdeckung zu erreichen.

Aufgabe 2 Modifizierter Bedingungs-/ Entscheidungsüberdeckungstest

Betrachten Sie die folgende, in Pseudo-Code angegebene, Bedingung. `A`, `B`, `C` und `D` stellen dabei atomare Bedingungen dar.

```
if (A || (B && C) || (B && D)){
        // if code
} else{
        // else code
}
```

(a) Erstellen Sie für die gegebene Bedingung die vollständige Wahrheitstabelle. Gehen Sie dabei von einer vollständigen Auswertung durch den Compiler aus.

(b) Geben Sie eine minimale Anzahl an Testfällen an, um den Modifizierten Bedingungs-/ Entscheidungsüberdeckungstest vollständig zu erfüllen. Markieren Sie die Testfälle in der Wahrheitstabelle und machen Sie deutlich, welcher Testfall für welche atomare Bedingung nötig ist.

(c) Gehen Sie nun von einer unvollständigen Auswertung durch den Compiler (von links nach rechts, unter Berücksichtigung der Schachtelung) aus. Erstellen Sie für diesen Fall eine neue Wahrheitstabelle.

(d) Wählen Sie für diese neue Wahrheitstabelle Testfälle aus, die den Modifizierten Bedingungs-/ Entscheidungsüberdeckungstest vollständig erfüllen. Markieren Sie erneut die Testfälle in der Wahrheitstabelle und machen Sie deutlich, welcher Testfall für welche atomare Bedingung nötig ist.

Aufgabe 3 Bounded-Interior-Testkriterium

Gegeben ist die in Java implementierte Funktion fct:

```
1  public int fct(List<Integer> list) {
2          int diff = 0;
3          for (Integer number : list) {
4                  if (number < 0) {
5                          diff--;
6                  } else if (number == 0) {
7                          break;
8                  } else {
9                          diff++;
10                 }
11         }
12         return diff;
13 }
```

(a) Erstellen Sie einen Kontrollflussgraphen für die Funktion fct.

(b) Ermitteln Sie für die Funktion fct Testfälle für die Erfüllung des Boundary-Interior-Kriteriums. Geben Sie jeweils den Eingabewert für $list$, den Rückgabewert und den durchlaufenen Pfad an. Auch das Erreichen des Break-Befehls zählt dabei als ein Schleifendurchlauf.

(c) Wir wollen zeigen, dass in der Praxis nicht immer alle Fälle des Bounded-Interior-Testkriteriums erfüllbar sind. Machen Sie im Code **vor** der Schleife eine Ergänzung, so dass die Schleife — wenn sie erreicht wird(!) — immer mindestens einmal durchlaufen wird. Die Funktionalität der Methode darf dabei nicht verändert werden!

(d) Für Schleifen der Form **do{C}while(B)** (mit C als Codeblock und B als Bedingung) lässt sich das Bounded-Interior-Testkriterium nicht direkt anwenden. Beschreiben Sie das Problem und benennen Sie kurz zwei Lösungsansätze, um das Problem zu lösen.

Hinweis: Denken Sie an Ansätze, die Änderungen des Codes oder des Testkriteriums betreffen.

Hinweise zu den Übungen:

- Durch die Teilnahme am Übungsbetrieb können Sie sich bis zu **3 Bonuspunkte für die Klausur** verdienen.
- Bedingungen:
  - Während des Semesters darf maximal eine Abgabe im ILIAS ausgelassen werden.
  - Während des Semesters darf pro Person max. eine Hörsaalübung ausgelassen werden.
  - Jede Gruppe muss im Laufe des Semesters eine Aufgabe in der Hörsaalübung präsentieren.
  - Jede Präsentation wird mit folgender Skala bewertet:
    - 3 Punkte: Korrekt
    - 2 Punkt: Sinnvoll aber fehlerhaft
    - 1 Punkte: Sinnlos/falsch oder fehlt (Basispunkt für Präsentation)

Viel Erfolg!