Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing-Wiss. und Informatik
- Software-Entwicklung -

Software-Qualität

Sommersemester 2026
5. Lernnotiz

Hinweis: Diese Lernnotiz enthält einen sehr sinnvollen Vorschlag um den Lehrstoff der 5. Woche der Veranstaltung (am 9.4) zu erlernen. Er ist gegliedert in die generellen Ziele und die Arbeitsschritte. Es ist notwendig, dass Sie die in dieser Lernnotiz genannten Videos bis zum Ende der offiziellen Vorlesungszeit (Do 18:00) durchgearbeitet haben. Zur Vorlesungszeit besteht die Möglichkeit in Zoom Fragen zu stellen und weitergehende Themen zu diskutieren.
https://hs-osnabrueck.zoom.us/my/kleuker

Ziele

- Selbständige Fähigkeit zum Einsatz von Kontrollflussgraphen für verschiedene Überdeckungen erwerben.
- Unterschiedliche Varianten der Bedingungsüberdeckungen verstehen, sie anwenden und gegeneinander abgrenzen können.
- Fähigkeit zur selbstständigen Erstellung und Auswertung von Datenflussgraphen zur Bestimmung der Effekte von Variablenänderungen.
- Im tiefsten Detail verstehen, warum Überdeckungsmessungen für die Software-Qualität notwendig sind, aber bei Weitem nicht alle Probleme entdecken müssen.

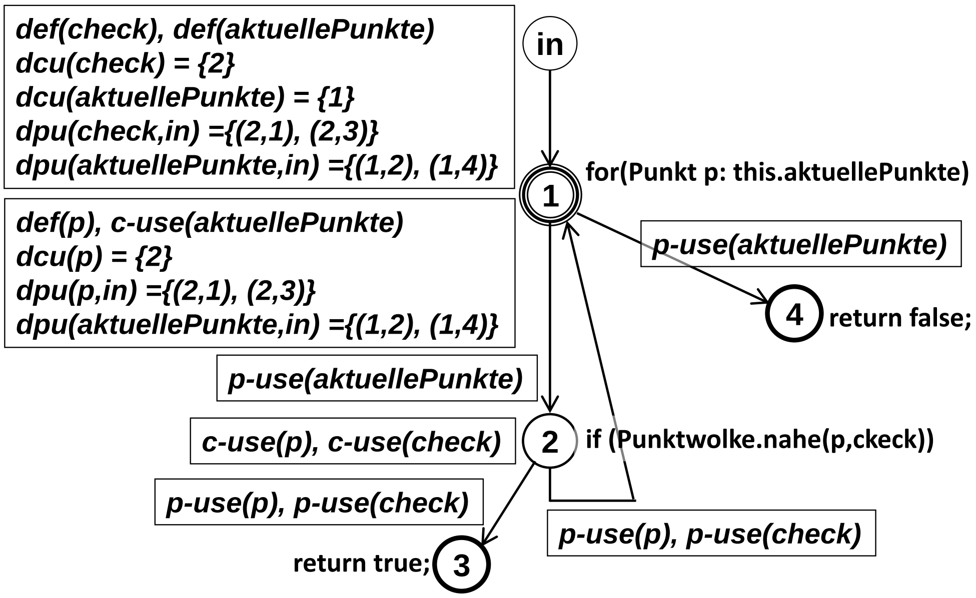
Arbeitsschritte

- *Laden Sie sich die folgenden Videos zuerst herunter, wenn Sie die HS-Plattform nutzen* und schauen Sie sich diese an. Es ist sinnvoll die Folien danach nochmals durchzugehen.

  Folien 156 – 208:
  http://kleuker.iui.hs-osnabrueck.de/Videos/QS/SQ_VL05_Testueberdeckungen.mp4
  (90:38), auch https://youtu.be/JL2HXRFreGg

  Die Betrachtungen in der Vorlesung fokussieren auf Methoden ohne die explizite Nutzung von Klassen, Objekten oder Objektvariablen, was hier in einem weiteren Beispiel passiert. Der Ansatz ist einfach zu erweitern, es wird aber auch deutlich, welcher Testaufwand entstehen kann. Dazu wird folgende einfache Methode betrachtet, der ein Punkt übergeben wird und die prüfen soll, ob es einen Punkt in der Liste this.aktuellePunkte gibt, der dem übergebenen Punkt nah ist. Die Überprüfung der Nähe findet in der Klassenmethode Punktwolke.nahe() statt.

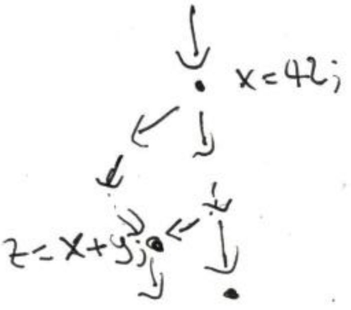
```
public boolean inDerNaehe(Punkt check){
    for(Punkt p: this.aktuellePunkte){
        if (Punktwolke.nahe(p,ckeck)){
          return true;
        }
    }
    return false;
}
```



Die Abbildung zeigt das Datenflussdiagramm mit den zugehörigen Einflussmengen. Zusätzlich zu den bekannten Betrachtungen fällt auf, dass der neue Startknoten mit def(aktuellePunkte) markiert ist, da innerhalb der Methode mit dieser Variablen gerechnet wird. Generell würde hier auch die Klasse Punktwolke genannt werden, da sie ebenfalls in einer Berechnung vorkommt. Da hier angenommen wird, dass die Klasse beim Aufruf dieser Methode nicht verändert wird, ist sie im konkreten Fall nicht angegeben. Weiterhin zeigt die Abbildung, dass eine Schleife wie ein if markiert wird und dass ein mit if markierter Knoten bei Methodenaufrufen auch c-use-Markierungen haben kann.

Falls jemand noch eine alternative Erklärung der Datenflussanalyse haben möchte, können der folgende Text hilfreich sein:

Grundidee Datenflussanalyse:
an welchen Stellen hat ein Befehl Auswirkungen



wann hat dieser Befehl eine Auswirkung auf z=x+y;
-> es gibt einen Pfad, auf dem x nicht verändert wird
(gibt es auf dem Pfad x=x+1; hat x=42; Auswirkung auf diese Anweisung und dann x=x+1; Auswirkung auf z=x+y;
d.h. es werden nur direkte Auswirkungen betrachtet)

Abstraktion durch Knotenmarkierungen:
def(x): hier wird x gesetzt (definiert)
c-use(x): hier wird x in einer Berechnung genutzt, (computanial use)

Bsp.: x++; x wird zur Berechnung genutzt: c-use(x)
x wird neu gesetzt (erhöht): def(x)

p-use(x): x wird in einem Booleschen Ausdruck genutzt, z.B. x>0 (predicative use)

Die Berechnung der möglicherweise betroffenen Anweisungen erfolgt mit dcu(x,n), im Knoten n wird x definiert und es werden alle Knoten benannt,

in denen die Zuweisung an x Einfluss haben könnte.
D.h. es gibt einen Pfad zu diesem Knoten, in dem x in einer Berechnung genutzt wird, auf dem x nicht verändert wird.

dpu(x,n) ähnlich definiert, nur Kanten in denen x bei einer Entscheidung genutzt wird.

Die zugehörigen Testüberdeckungen prüfen, ob diese Nutzungen von x auch alle in den Tests berücksichtigt werden.

all c-uses: Wenn eine Definition def(x) im Knoten n passiert und dass einen Effekt auf einen Knoten m hat, muss es einen Test geben, der von n zu m läuft, auf dem zwischenzeitlich x nicht geändert wird

all p-uses: analog zu den Kanten, bei denen x Einfluss hat

Generell gewünscht ist dann, dass all c-uses und all p-uses erreicht wird.

Die Überdeckung all-defs ist eine sehr schwache Variante in der nur gefordert wird, dass alle so markierten Knoten erreicht werden. Anschaulich: Jede Zuweisung wird einmal durchlaufen.

- Lesen Sie zur Wiederholung und Vertiefung in [Kle26] die Seiten 141-163.
- Bearbeiten Sie das Quiz unter http://kleuker.iui.hs-osnabrueck.de/quiz/qs05_23357.html und merken Sie sich die oben angegebenen Lösungsbuchstaben.
- Bearbeiten Sie Aufgabenblatt 5. Denken Sie daran, dass ich für Fragen meist kurzfristig erreichbar bin.

- Lesen Sie das zur Vorlesung gehörende Fragen-Und-Antworten-Dokument, das meist kurz nach der Vorlesung auf der Veranstaltungsseite in der Nähe dieser Lernnotiz steht.

- Prüfen Sie, ob Sie die angegebenen Lernziele erreicht haben.