12. Übungsblatt zu Software Qualität

Michel Meyer, Manuel Schwarz

23. Januar 2013

Aufgabe 12.1 - Anomalieanalyse

(a) Java-Beispiele

Schnittstellenanomalie

Listing 1: Schnittstellenanomalie

```
// Methodenkopf mit bestimmter Parameterliste
public boolean binSearch(int[] a, int n){
  // some code...
}

// Methodenaufruf
int[] numbers = new int[100];
int n = 42;
boolean found = binSearch(n, numbers);
```

Die Schnittstellenanomalie wird vom Compiler erkannt und erzeugt einen Fehler.

Variablendeklaration-/-nutzungsanomalie

Listing 2: Variablendeklarations-/-nutzungsanomalie

```
int n = 42;
String s = "test";
int result = n + s;
```

Die Variablendeklarations-/-nutzungsanomalie führt ebenfall zu einem Fehler, der durch den Compiler erkannt wird.

Kontrollflussanomalie

Listing 3: Kontrollflussanomalie

```
public boolean binSearch(int[] a, int n){
  return null;
  if(a.contains(n)){ // der hier folgende Code wird niemals erreicht
    // do stuff
  }
}
```

Die Kontrollflussanomalie führt ebenfalls (in Java) zu einem Fehler, falls der Code nicht erreichbar ist.

Datenflussanomalie

Listing 4: Datenflussanomalie

```
// Beispiel 1
int a = 3;
int a = 7; // dd-Anomalie, kein Fehler

// Beispiel 2
int a;
int b = 5 + a; // ur-Anomalie, Fehler
```

Die Datenflussanomalie kann (in Java) zu einem Fehler führen, wenn eine Variable beispielsweise nicht initialisiert wurde, bevor mit ihr gerechnet wird. Doppelzuweisungen führen dagegen zu keinem Fehler.

Aufgabe 12.2 - Model Checking

(a) Berechnungsbaum

Abbildung 1: Berechnungsbaum