ToiletPaper #132

Mutationtests mit Pitest

Autor: Christof Huber / Software Developer / Business Division Automotive Bavaria

✗ Problem

Übliche Testmetriken, wie z.B. line coverage und branch coverage, geben lediglich darüber Auskunft, ob Code bei Testausführung erreicht worden ist und nicht unbedingt, ob auch die richtige Funktionalität ausgeführt worden ist.
Dadurch ist es möglich Testcode zu schreiben, der lediglich die Metrik zufriedenstellt, ohne irgendetwas zu testen (Testcases ohne Assertions).

✓ Lösung

Um sich über die Qualität der Tests ein Bild zu machen, kann man Mutationtests verwenden.
Ein Mutationtest führt Unittests auf verschiedenen Mutationen des zu testenden Codes aus.
Es gibt viele verschiedene Mutationsarten, z.B. das Ändern von Conditions (>= wird zu >) oder das Invertieren von boolschen Rückgabewerten.

➔ Beispiel

```
public class Jambitee {
  public boolean isAwake(int amountOfCoffeesToday) {
    if (amountOfCoffeesToday >= 1 ) {
      return true;
    } else {
      return false;
    }
  }
}

public class JambiteeTest {
  private Jambitee jambitee = new Jambitee();

  @Test
  public void isAwakeWith2Coffees() {
    assertTrue(jambitee.isAwake(2));
  }

  @Test
  public void isAsleepWithNoCoffee() {
    assertFalse(jambitee.isAwake(0));
  }
}
```

→ Obwohl der Code hier eine 100%ige line- und branch-coverage hat, sichern die Tests die ">=1"-Condition nicht ab.

In Maven muss man nur das Plugin einbinden.

```
<plugin>
  <groupId>org.pitest</groupId>
  <artifactId>pitest-maven</artifactId>
  <version>LATEST</version>
</plugin>
```

Danach kann man die Mutationtests ausführen, indem man das mutationCoverage Goal aufruft.
`mvn org.pitest:pitest-maven:mutationCoverage`

Jambitee.java

```
1  package com.jambit.chuber.toiletpaper;
2
3  /**
4   * @author Christof Huber
5   * @version 25.06.20
6   */
7  public class Jambitee {
8
9      public boolean isAwake(int amountOfCoffeesToday) {
10 2        if (amountOfCoffeesToday >= 1) {
11 1            return true;
12         } else {
13 1            return false;
14         }
15     }
16 }
```

Mutations

```
10 1. changed conditional boundary → SURVIVED
   2. negated conditional → KILLED
11 1. replaced boolean return with false for com/jambit/chuber/toiletpaper/Jambitee::isAwake → KILLED
13 1. replaced boolean return with true for com/jambit/chuber/toiletpaper/Jambitee::isAwake → KILLED
```

→ Pitest führt die Tests auf verschiedenen Mutationen aus und erkennt dabei, dass es eine Mutation gibt, die nicht von den Tests abgedeckt ist.

✚ Weiterführende Aspekte

- https://pitest.org/
- https://pitest.org/quickstart
- https://github.com/hcoles/pitest