Test Driven Development powered by KI

GEBIT SOLUTIONS
Retailers First Choice

Nicolai Mainiero

Motivation

- ✔ Spezifikation durch RE
- ✔ Implementierung durch Developer
- ✔ Evtl. Tests durch Developer/QA

- ✔ Aktueller Trend, Tests durch LLMs erzeugen lassen

➔ Tests prüfen Implementierung, nicht Spezifikation

Idee

- ✔ Spezifikation durch RE
- ✔ Automatisierte Tests durch Developer/QA
- ✔ Implementierung durch Developer

➔ Test-driven Development

Testing Basics

Unit-Test

Beispiel

- Reduziert Kosten
- Gibt Sicherheit beim Refactoring
- Verhindert bereits früh Fehler

- Modul wird Isoliert getestet
- KPIs führen oft zu White-Box-Testing
- Test kann selben Fehler wie Implementierung enthalten

```java
public class MyFirstJUnitTests {
    private final Calculator calculator = new Calculator();

    @Test
    void addition() {
        assertEquals(2, calculator.add(1, 1));
    }

    @ParameterizedTest
    @CsvSource(value = {"2:1:1", "4:2:2", "4:1:3", "4:0:4", "4:3:1", "4:4:0"},
               delimiter = ':')
    void addition(int expected, int a, int b) {
        assertEquals(expected, calculator.add(a, b));
    }
}
```

Property-based Test

Beispiel

- Testet vollständigen Gültigkeitsbereich
- Testet Verhalten (Black-Box)
- Erreicht hohe Testabdeckung

- Modul wird Isoliert getestet
- Manchmal schwierig geeignete Eigenschaften zu finden

```java
public class MyFirstPropertyBasedTest {
    private final Calculator calculator = new Calculator();

    @Property
    @Label("Addition is commutative")
    void additionIsCommutative(@ForAll int a, @ForAll int b) {
        assertThat(calculator.add(a, b)).isEqualTo(calculator.add(b, a));
    }

    @Property
    @Label("Addition is associative")
    void additionIsAssociative(@ForAll int a, @ForAll int b, @ForAll int c) {
        assertThat(calculator.add(calculator.add(a, b), c))
                .isEqualTo(calculator.add(a, calculator.add(b, c)));
    }

    @Property
    @Label("Addition with zero")
    void additionWithZero(@ForAll int a) {
        assertThat(calculator.add(a, 0)).isEqualTo(a);
    }
}
```

Behavior-Driven Development

Beispiel

- Natürlichsprachliche Beschreibung des erwarteten Verhaltens
- Nicht auf Module beschränkt

- Test muss in eigener Sprache formuliert werden
- Extra Code notwendig, um die Szenarien zu interpretieren

```
Feature: Calculator Addition

  As a user
  I want to use a calculator
  So that I can perform addition operations

  Scenario: Basic addition
    Given I have a calculator
    When I add 1 and 1
    Then the result should be 2

  Scenario Outline: Addition with various numbers
    Given I have a calculator
    When I add <a> and <b>
    Then the result should be <expected>

    Examples:
      | a   | b   | expected |
      | 2   | 1   | 3        |
      | 4   | 2   | 6        |
      | 4   | 1   | 5        |
      | 4   | 0   | 4        |
```

Mutation Testing

Beispiel

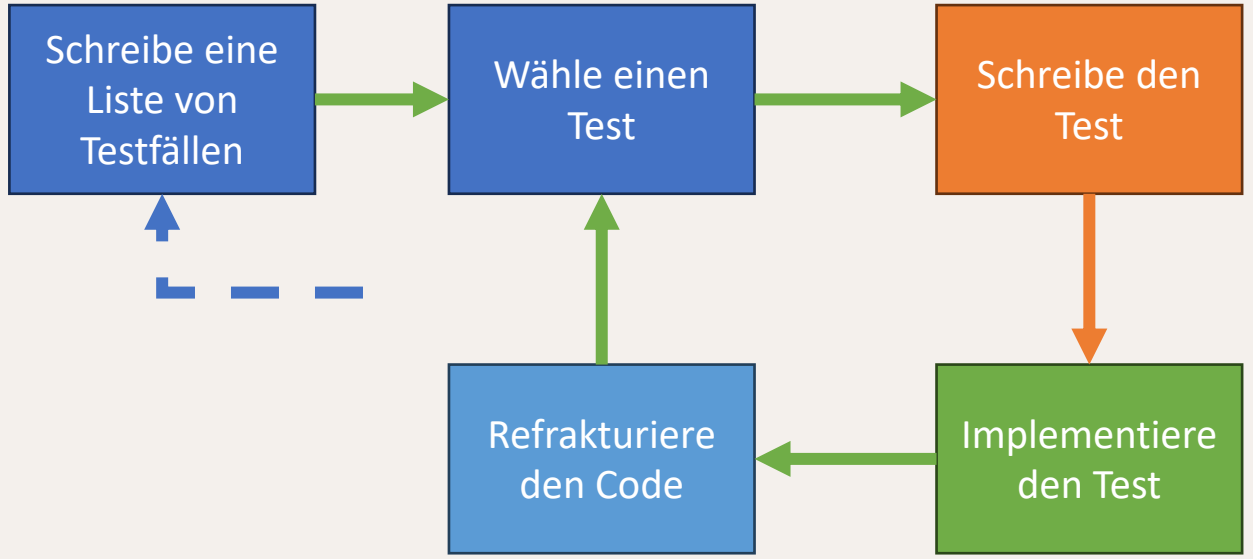
- Versucht bestehende Tests zu brechen indem Fehler in die Implementierung eingestreut werden
- Tests müssen alle mutierten Codevarianten identifizieren, sonst sind die Tests nicht gut genug

```
public class Calculator {

  public Integer add(int i, int j) {
    return i + j;
  }
}

Mutations
1. Replaced integer addition with subtraction → KILLED
2. replaced Integer return value with 0 for
   de/mainiero/Calculator::add → KILLED
```

Test-driven Development

Red – Green – Refactor – Cycle

- Schreibe einen neuen/nächsten Test für neue Funktionalität
- Implementiere den einfachsten Code, bis der Test bestanden ist
- Refrakturiere den neuen und bestehenden Code, so dass er gut strukturiert und verständlich ist



DEMO

Context macht den Unterschied

Auf dem Parkplatz vor dem Kino stehen 12 Fahrzeuge – Motorräder und Autos. Julian zählt die Reifen der Fahrzeuge: es sind insgesamt 32 Reifen. Wie viele Autos und Motorräder stehen auf dem Parkplatz?

Kreuze die richtige Lösung an.
Auf dem Parkplatz stehen:

- ☐ 8 Autos und 4 Motorräder
- ☐ 4 Autos und 7 Motorräder
- ☐ 6 Autos und 4 Motorräder
- ☐ 5 Autos und 7 Motorräder
- ☐ 4 Autos und 8 Motorräder
- ☐ 3 Autos und 8 Motorräder

Fazit

Fragen?