

## Softwaretechnik-Praktikum SS 2005

GR-1

Projektleiter: Adrian Kiess | Dokument erstellt von Guangyu Wang

## Testkonzept

2 TESTPROZESS

### Zusammenfassung

Das Testkonzept definiert die Abgrenzungskriterien zur Behebung von Mängeln der Software.

## 1 Einführung

Das systematische Testen von Software ist ein essenzieller Bestandteil des Entwicklungsprozesses. Der Testprozess soll die Korrektheit der Software nachweisen und sicherstellen.

Die Softwaretests werden mittels des JUnit-Framework<sup>1</sup> umgesetzt.

Der Testprozess wird in vier Bereiche gegliedert:

- Komponententest
- Integrationstest
- Systemtest
- Abnahmetest

Der für die Tests Verantwortliche<sup>2</sup>, hilft und koordiniert jedes Programmiererpärchen; damit für den erstellten Quellcode eine komplette *TestSuite*<sup>3</sup> zur Verfügung steht.

### 1.1 JUnit

JUnit bietet Werkzeuge, um Tests für Java-Quellcode zu implementieren. Die wichtigste Klasse des JUnit-Frameworks ist die Klasse *TestCase*, welche die Funktionalitäten zur Implementierung und zum Ausführen der Tests anbietet. Durch Implementierung von *TestCase* und Hinzufügen der Testmethoden, kann die Software getestet werden. Um einen Test auszuführen, wird der *TestCase* einer *TestSuite* hinzugefügt und die Methode *TestSuite.run()* aufgerufen. Die *TestSuite* enthält mehrere Testfälle, welche alle ausgeführt werden.

## 2 Testprozess

### 2.1 Komponententest

Bei einem Unit-Test handelt es sich um ein Stückchen Quellcode, das der Programmierer zum Testen eines sehr kleinen und klar umrissenen Teiles der Funktionalität der Software geschrieben hat. Mit der erfolgreichen Ausführung des Unit-Tests weist der Entwickler dessen korrekte Funktionalität nach. Die Testklassen werden vor der Implementierung der eigentliche Klasse erstellt, die Funktionalität der Klasse prädefiniert. Jede Methode der Klasse muss in der Testklasse durch ein vorangestelltes *test* im Methodennamen implementiert werden.

Eine typischer Testfall besteht aus drei Schritten:

- Testobjekt erzeugen

---

<sup>1</sup>JUnit Framework: <http://www.junit.org/>

<sup>2</sup>Verantwortlicher für die Tests: Guangyu Wang

<sup>3</sup>API-Beschreibung *JUnit-Framework-TestSuite*: <http://junit.sourceforge.net/javadoc/junit/framework/TestSuite.html>

## Softwaretechnik-Praktikum SS 2005

GR-1

Projektleiter: Adrian Kiess | Dokument erstellt von Guangyu Wang

### 2.2 Integrationstest

### 3 TESTBEISPIEL

- Methode vom Testobjekt ausführen
- Ergebnisse überprüfen

## 2.2 Integrationstest

Nachdem der Komponententest vollzogen ist, folgt der Integrationstest. Der Integrationstest ist die Erweiterung des Komponententest.

Es wird nach Fehlern in der Kommunikation und Zusammenspiel der Komponenten gesucht. Aus zwei getesteten Komponenten wird eine Gruppe gebildet - nun wird also das Interface der Komponenten getestet.

## 2.3 Systemtest

Das Ziel des Systemtestes ist die Überprüfung des Gesamtsystems auf Erfüllung der erwarteten Anforderungen. Beim Systemtest wird die Erfüllung sowohl von funktionalen wie auch nicht-funktionalen Anforderungen geprüft. Die wichtigste funktionale Anforderung ist sicherlich die Richtigkeit der errechneten Ergebnisse. Darunter fallen zum Beispiel: Sicherheit, Angemessenheit, Ordnungsmäßigkeit und Interoperabilität. Der nicht-funktionale Test kann folgende Anforderungen besitzen: Performanztest, Volumentest und Stresstest.

## 2.4 Abnahmetest

Der Abnahmetest ist der Test für das Endprodukt, er wird vom Auftraggeber durchgeführt.

## 3 Testbeispiel

Wir erzeugen eine Klasse, die zwei Nummern multipliziert und das Ergebnis zurückgibt.

```
public class Rechnen {
    public int multiplizieren(
        int ersterWert, int zweiterWert ) {
        return ersterWert * zweiterWert;
    }
}
```

Ein JUnit-Test zum Testen der *multiplizieren*-Methode in der *Rechnen*-Klasse, sähe wie folgt aus:

```
public class TestRechnen extends TestCase {
    public void testMultiplizieren {
        int ersterWert = 6;
        int zweiterWert = 2;
        int erwartetesErgebnis = 12; // 2x6=12 x-D

        Rechnen r = new Rechnen();
        int ergebnis =
            r.multiplizieren(ersterWert, zweiterWert);
        assertTrue( erwartetesErgebnis == ergebnis );
    }
}
```

## Softwaretechnik-Praktikum SS 2005

GR-1

Projektleiter: Adrian Kiess | Dokument erstellt von Guangyu Wang

---

*3 TESTBEISPIEL*

Nun können wir den Test ausführen, indem wir eine Instanz der *TestRechnen*-Klasse erzeugen, und die Methode *testMultiplizieren* ausführen.

```
TestCase test = new TestRechnen("testMultiplizieren");
test.run();
```

Sollten die Ergebnisse nicht übereinstimmen, wird JUnit eine Exception werfen.