Technische Universität Berlin

Grundlagen des Software-Testens
Übung #2 – Komponenten- und Integrationstest

Prof. Dr. Ina Schieferdecker, Theofanis Vassiliou-Gioles, Julia Martini

Quality Engineering of Open Distributed Systems

QDS Quality Engineering of Open Distributed Systems

Vorbereitung

Technische Universität Berlin

Ziel der Übung

- Verständnis der **Stufen des Testprozesses**
- Durchführung eines **Komponententest**s mit Mock-Up Objekten und JUnit 4
- Durchführung eines **Integrationstest**s mit JUnit 4
- Erstellen von **TestSuite**s mit JUnit 4

Abgabe der Übungsaufgabe 2

- Bis zum Anfang der nächsten Vorlesung

QDS Quality Engineering of Open Distributed Systems

MVC Entwurfsmuster

Model

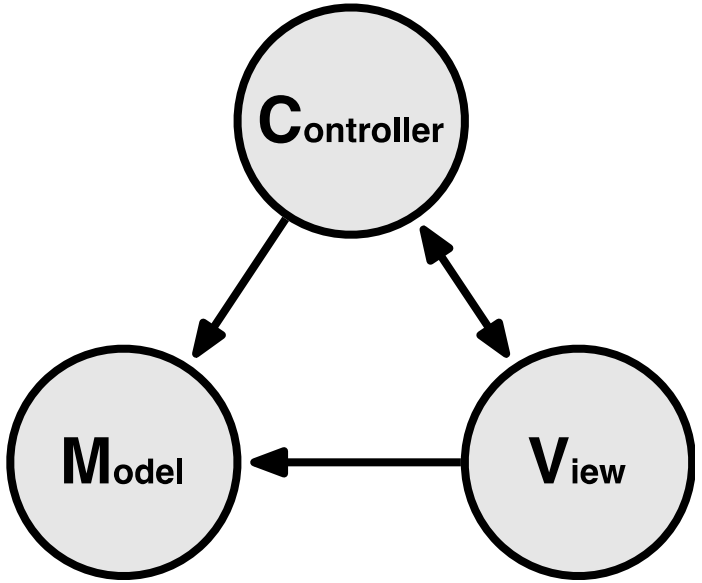
- Datenhaltung

View

- Datenpräsentation
- User Interaktion

Controller

- Koordiniert Model und View
- Wertet Aktionen aus



JUnit 4 TestSuites

Eine TestSuite fasst eine Menge JUnit Testklassen zusammen und ermöglicht eine automatisierte Ausführung mehrerer Testklassen

```
@RunWith(Suite.class)
@SuiteClasses( {foo.class, bar.class} )
public class ExampleSuite {
}
```

Unit Tests und Ausnahmen

Tipp:
Testen Sie Ausnahmen mit try/catch Konstrukten

```
try {
    someObject.someMethod(foo);
    fail();
} catch (SomeException e) {
    // OK, expected behaviour
}
```

Mock-Up Klassen

Stellen Platzhalter für eine Komponente dar ohne die Funktionalität der Komponente vollständig nachzubilden

Werden oft mit zusätzlicher Testinstrumentierung versehen um bestimmte Verhaltensmuster absichtlich herbeizuführen

- Beispiel: Werfen einer Ausnahme

Besitzen syntaktische Konformität mit der nachzubildenden Komponente

- Signatur der Schnittstelle

Verwendung von Mock-Up Frameworks möglich (EasyMock, PowerMock, ...), aber nicht notwendig.

Theoretische Aufgaben

1. Sie befinden sich im Testlabor eines Navigationsgeräte-Herstellers und beobachten ein Team, das ein neues Navigationsgeräte-Modell auf einem Prüfstand zur Emulation von Fahrzeugbewegungen mit unterschiedlichen Interaktionsfolgen bedient und prüft.
   In welchen Testphasen kann sich das beobachtete Team befinden? Begründen Sie Ihre Festlegung.

2. Sie möchten Ihren Vorgesetzten/Ihre Vorgesetzte für Unterstützung für einen modernisierten Testprozess gewinnen, bei dem das Testen frühzeitig mit den Entwicklungsaktivitäten verzahnt wird.

   Was schlagen Sie konkret für die Modernisierung eines bis dato auf dem Wasserfallmodell basierten Testprozesses vor? Bitte nennen Sie drei nötige Änderungen und erläutern diese kurz.
   Welche Argumente würden Sie vorbringen, um Ihren Vorgesetzten/Ihre Vorgesetzte vom Zweck der vorgeschlagenen Änderungen zu überzeugen. Nennen Sie drei Argumente und erläutern diese kurz.

Praktische Aufgaben

3. Testen Sie im Rahmen eines Komponententests der Klasse `AddressBookControllerImpl` die Methode `add(...)`.
   - Schreiben Sie für die Model und View Komponenten Mock-Up Klassen und verwenden Sie diese im Komponententest.
   - Testen Sie gründlich - es sind Fehler zu finden.

4. Programmieren Sie einen Integrationstest für `AddressBookModel` und `AddressBookController`.
   - Testen Sie, ob die Methoden des `AddressBookController` Interface zu den erwarteten Resultaten im Addressbuch führen.
   - Testen Sie intensiv und schreiben Sie **MINDESTENS** einen Testfall pro Methode des Interfaces. Es sind Fehler zu finden.

5. Erstellen Sie eine JUnit-TestSuite, mit der Komponenten- und Integrationstest automatisch ausgeführt werden können

Vorgehensweise praktische Aufgaben

1. Eclipse starten und das Übungsprojekt importieren
   - Das Übungsprojekt findet sich als Anhang **project2.zip**
   - Das Beispiel verstehen
     - Die Beschreibung zum `AddressBookController` lesen
     - Den Quellkode anschauen
     - Die Klasse **exercise2.addressbook.Manager** ausführen
2. Dateien für die Mock-Ups bearbeiten
   - /Uebung2/src/exercise2/test/***MockUp.java**
3. TestSuite erstellen
   - Komponententest des Controller: **AddressBookControllerTest.java**
   - Integrationstest für Controller und ModelController: **AddressBookIntegrationTest.java**
   - Eine **TestSuite**, die beide enthält (Datei nicht vorgegeben)
4. Einige Testfälle sollen Fehler finden!
5. Die fertigen Testklassen abgeben
   - Zusammen mit den theoretischen Aufgaben in einer ZIP Datei