Freie Universität Berlin

Grundlagen des Software-Testens

Übung #2

Komponenten- und Integrationstest

Ina Schieferdecker, Edzard Höfig
Fachgruppe Modellbasierte Entwicklung und Qualitätssicherung
Arbeitsgruppe Software Engineering
Institut für Informatik

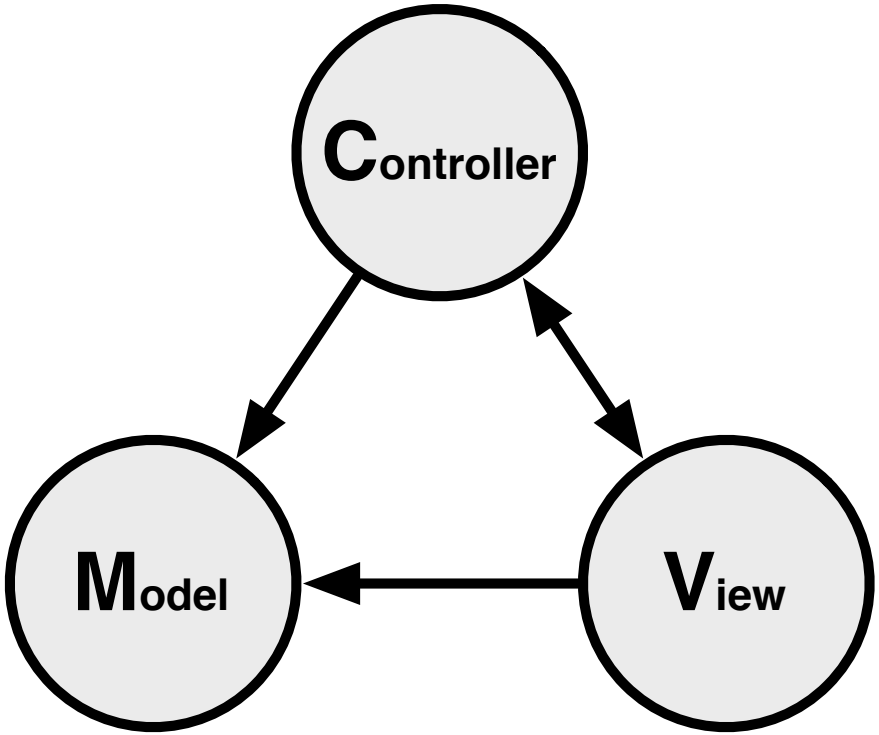
Vorbereitung

- 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 Übung**
    - Deadline ist am nächsten Freitag, dem 24.08.2012 um 14h
  - In **Blackboard**
  - **Pro Gruppe** eine Abgabe

MVC Entwurfsmuster

- Model
  - Datenhaltung
- View
  - Datenpräsentation
  - User Interaktion
- Controller
  - Koordiniert Model und View
  - Wertet Aktionen aus



JUnit 4 TestSuites

Freie Universität Berlin

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 {
}
```

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

Theoretische Aufgabe 1

Freie Universität Berlin

Nennen Sie sechs Faktoren, die die Qualitätssicherung heutiger software-basierter Systeme anspruchsvoll machen.

Theoretische Aufgabe 2

Erläutern Sie die Unterschiede zwischen interner und externer Software-Qualität nach ISO 9126 in einer vergleichenden Analyse eines Merkmals für interne und für externe Qualität.

Praktische Aufgaben

Freie Universität Berlin

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

Freie Universität Berlin

1. Eclipse starten und das Übungsprojekt importieren
   - Das Übungsprojekt findet sich als Anhang **projekt2.zip** in Blackboard (im Gruppenabschnitt)
2. Das Beispiel verstehen
   - Die Beschreibung zum `AddressBookController` lesen
   - Den Quellkode anschauen
   - Dazu die Klasse **exercise2.addressbook.Manager** ausführen
3. Dateien für die Mock-Ups bearbeiten
   - /Uebung2/src/exercise2/test/***MockUp.java**
4. TestSuite erstellen
   - Komponententest des Controller: **AddressBookControllerTest.java**
   - Integrationstest für Controller und ModelController: **AddressBookIntegrationTest.java**
   - Eine **TestSuite**, die beide enthält (Datei nicht vorgegeben)
5. Einige Testfälle sollen Fehler finden!
6. Die fertigen Testklassen per Blackboard abgeben
   - Zusammen mit den theoretischen Aufgaben