Technische Universität Berlin

Grundlagen des Software-Testens
Lösungen zur Übung #2

Prof. Dr. Ina Schieferdecker, Theofanis Vassiliou-Gioles, Julia Martini
Quality Engineering of Open Distributed Systems

QDS Quality Engineering of Open Distributed Systems

Aufgabe 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 Teststufen kann sich das beobachtete Team befinden? Begründen Sie Ihre Festlegung.

Integrationstest oder Systemtest, aber weder Komponententest noch Abnahmetest

Aufgabe 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.

Aufgabe 2) Vorschläge und Argumente

Tester begleiten die Anforderungs- und Entwurfsphase

- Frühes Testen senkt Fehlerfolgekosten
- Frühes Testen erlaubt einen zeitnahen kostengünstigen Fix in Anforderungs- bzw. Entwurfsspezifikationen

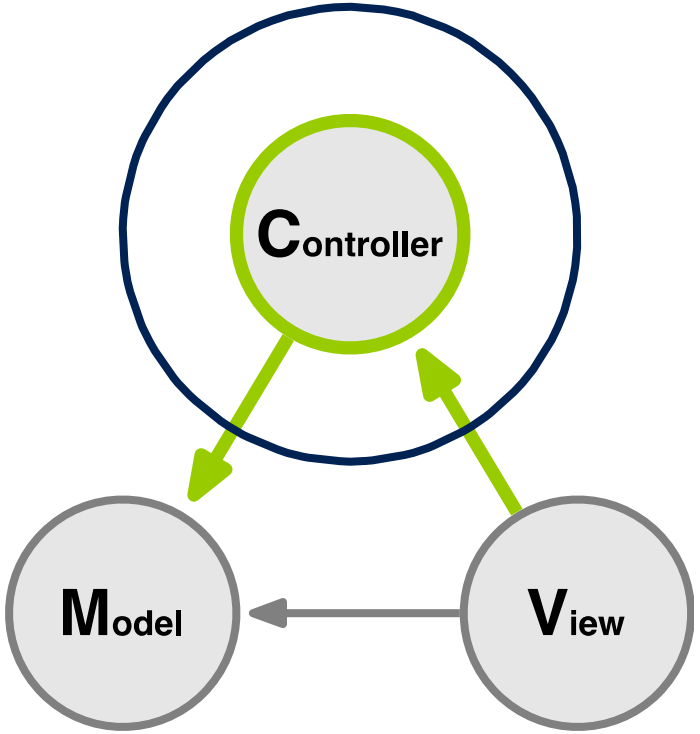
Tests werden parallel zur Software entwickelt

- Tests stehen mit der Verfügbarkeit erster Software-Komponenten bereit

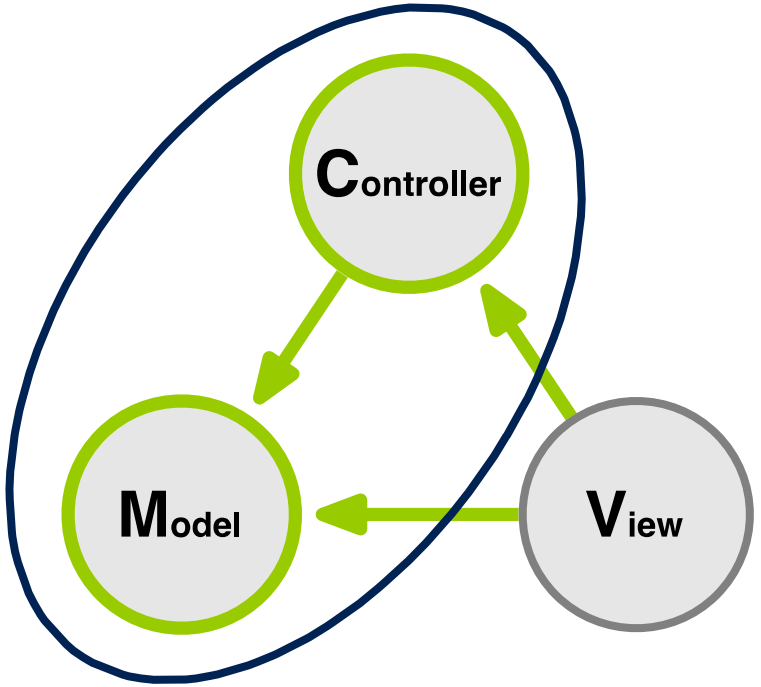
Tests werden iterativ bei Verfügbarkeit erster Komponenten realisiert

- Tester arbeiten sich zeitnah in die zu entwickelnde Software und die genutzten Technologien ein

Aufgabe 3) Komponententest



Aufgabe 3) Integrationstest



Aufgabe 3a)

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 dies im Komponententest.
- Testen Sie gründlich - es sind Fehler zu finden!

Fehlerfälle:

Beide Parameter `PhoneContactInformation` und `EmailContactInformation` sind bei Aufruf auf `null` gesetzt

- Es wird keine `ParameterException` geworfen

Der `gender` Parameter ist `null`

- Es kommt zu einer `NullPointerException` in der Methode

Es wird keine `SizeLimitReachedExeption` geworfen, wenn das `AddressBook` voll ist

Aufgabe 3b)

Programmieren Sie einen Integrationstest für AddressBookModel und AddressBookController.

- Testen Sie ob die Methoden des AddressBookController Interface zu den erwarteten Resultate im Addressbuch führen. Testen Sie intensiv und schreiben Sie MINDESTENS einen Testfall pro Methode des interfaces. Es sind Fehler zu finden.

Fehlerfälle:

Es wird keine `SizeLimitReachedExeption` geworfen, wenn das `AddressBook` voll ist

Die `erase()` Methode des Controllers ist falsch implementiert und ruft nicht die korrekte Methode aus `AddressBook` auf

Aufgabe 3c)

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

```
@RunWith(Suite.class)
@SuiteClasses({
  AddressBookControllerTest.class,
  ControllerAddressBookIntegrationTest.class
})
public class AllTests {
}
```

Telefonnummern

Sind keine **`int`**

- **`+49 30 12345-67`** (DIN 5008)
- **`+49-30-1234567`** (URI konform)
- **`0049 (0)30 12345-67`** (auch möglich)

Besser: Felder mit 15 **`char`**

- Führende Nullen, bzw. **+** als Präfix
- Siehe ITU-T E.123 und E.164

Vorverarbeitung notwendig

- Normalisierung
- Verliert dann aber Informationen (Durchwahl)