

Unit Testing

Lukas Panni

TINF18B5

DHBW Karlsruhe

Vorlesung Advanced Software Engineering Semester 5/6

Inhaltsverzeichnis

1	Analyse und Begründung für Umfang der Tests	3
2	Analyse und Begründung für Einsatz von Fake-/Mock-Objekten	4

1 Analyse und Begründung für Umfang der Tests

Um die Funktionalität der einzelnen Komponenten gewährleisten zu können werden Unit-Tests eingesetzt. Dabei werden für die einzelnen Tests nur die, für diesen Test, relevanten Teile des Systems verwendet. Da Abhängigkeiten zu anderen Komponenten die Tests nicht beeinflussen sollen werden alle anderen Komponenten durch Fake-/Mock-Objekte ersetzt. Das Zusammenspiel mit den anderen Komponenten kann in Integrationstests getestet werden. Außerdem tragen Unit-Tests auch zur Dokumentation bei, indem das gewünschte Verhalten der Komponente für Regel- und Ausnahmefälle in den Testfällen dokumentiert ist.

Für dieses Projekt wird *JUnit* als Framework für die Erstellung und Ausführung von Java-Unit-Tests verwendet. Bei der Implementierung der Tests wurde darauf geachtet die ATRIP-Regeln (*Automatic, Thorough, Repeatable, Independent, Professional*) möglichst zu beachten.

Teile des Codes in diesem Projekt werden für Android-Spezifische UI-Aufgaben benötigt und können nur schlecht getestet werden. Das führt auch dazu, dass die Code-Coverage über das gesamte Projekt vergleichsweise klein sein kann. Stattdessen sollte hier zur Beurteilung der Testabdeckung nur die Code-Coverage der anderen Klassen betrachtet werden.

2 Analyse und Begründung für Einsatz von Fake-/Mock-Objekten

Fake- und Mock-Objekte werden benötigt, um Abhängigkeiten einer Komponente zu anderen Komponenten in Unit-Tests zu reduzieren. Sie implementieren dafür zum Beispiel das benötigte Interface, aber davon nur die aktuell benötigte Funktionalität.