# TU Wien:Software-Qualitätssicherung VU (Biffl)/Prüfung 2014-05-14

< TU Wien:Software-Qualitätssicherung VU (Biffl)

## Contents



## Qualitätsmanagement VS Qualitätssicherung, Prinzipien des Qualitätsmanagements nennen

**Qualitätssicherung:** Verfahren, Aktivitäten und Prozesse, welche sicherstellen, dass ein Produkt gegebenen Anforderungen (Spezifikation und Kundenanforderungen) gerecht wird.
**Qualitätsmanagement:** Gesamtheit aller Maßnahmen und Aktivitäten, die zur Herstellung und Wahrung von Qualität notwendig sind. Dazu zählen unter anderem Qualitätsplanung, -lenkung, -sicherung und insbesondere die Qualitätsverbesserung.

## Functionality, Usability, Reliablity, Performance und Supportability: Worauf muss der Kunde bei einem Akzeptanztest achten? Zwei Merkmale für jede dieser Anforderungen sind zu nennen.

**Akzeptanztest:** Der Kunde prüft hier die Einhaltung und Umsetzung seiner Anforderungen, welche im Idealfall der Projektspezifikation entsprechen (sonst kommt es wohl oder übelst zu Disputen).

**Functionality:** Dabei handelt es sich um Features des Produktes, z.B. die Fähigkeit, eine gegebene Aufgabe zu erfüllen. Grob gesagt jede Funktionalität, die explizit eingebaut wurde.
**Usability:** Behandelt die Art und Weise, wie (*einfach oder komplex*) mit dem System interagiert werden kann. Entsprechende Hilfeseiten und Dokumentation zählen auch zu diesem Punkt.
**Reliability:** Befasst sich mit der Verlässlichkeit des Produktes, bspw. wie oft das System ausfällt, wie einfach/schnell die Wiederherstellung (Recovery) funktioniert und wie vorhersehbar solche Ausfälle sind.
**Performance:** Ist ein wichtiges Merkmal für Betreiber, da bei guter Performance an Hardware gespart werden kann. Beachtet werden Durchsatz, Antwortzeiten und ggf. auch die Genauigkeit des Systems.
**Supportability:** Bezieht sich auf den Betrieb eines Systems, sprich wie gut es sich warten oder erweitern lässt. Auch die Konfiguration zählt hier dazu.

## Ziele von technischen Reviews, Inspektionen und Code-Walkthrough.

Grundsätzlich dienen diese Maßnahmen der Wahrung und Verbesserung der Qualität eines Produktes (d.h. Fehler früh finden und beheben). Sie können auch zur Kontrolle von Konsistenzen (vgl. Spezifikation vs. UML-Diagramme) herangezogen werden. Die Verbesserung des Entwicklungsprozesses ist ebenfalls ein Kernziel.

Oftmals werden Reviews und Inspektionen auch als Grundlage zur Beurteilung der Qualität eines Softwareproduktes verwendet.

## Black-Box Test VS White-Box Test. Ist es möglich White-Box Tests mit dem JUnit Framework durchzuführen. Wie? Begründung!

**Black-Box Tests:** Der Tester hat keine Kenntnis (oder ignoriert seine Kenntnis) über den Aufbau der zu testende Softwarekomponente. Der Test basiert daher rein auf der Spezifikation, sprich welche Ausgabe bei entsprechender Eingabe erwartet wird.
**White-Box Tests:** Bei dieser Testart hat der Tester Kenntnis über den Aufbau einer Komponente und nutzt dieses Wissen zum Schreiben der Tests aus. Es werden also Code-Teile, wie z.B. Bedinungsanweisungen, mit genau diesem Wissen im Hinterkopf überprüft ("Logik getriebene Tests"). Dadurch können verschiedene Codeabdeckungsarten erreicht werden (C0-C3). Es ist selbstverständlich möglich, mit JUnit White-Box Tests durchzuführen. Dazu schreibt man Tests, die beispielsweise alle Bedingungen oder Kanten in einer Komponente explizit durchlaufen.

## Kontrollflussgraph war gegeben. Testfälle bezüglich 100% Anweisungsüberdeckung und 100% Bedingungsüberdeckung waren gefragt.

- Multiple-Choice Frage

## Welche Code-Quality-Checks gibt es? Nennen Sie ein Tool das hierbei Unterstützung bietet.

- Checkstyle (hinsichtlich Uniformität und Konventionen, z.B. Kommentare, Variablenbenennung, Duplikation von Code, ...)
- Code-Coverage Checks (*Wie viel Produktionscode wird durch Unit Tests abgedeckt?*)
- Find-Bugs (Prüft auf Bug Patterns bzw. Bad Practice)

## Teststufen (Modul, Integration, Last, Regression, System, Akzeptanz) mit der richtigen Fällen (Erklärungen) verbinden.

- *Quasi* Multiple-Choice Frage

## Wozu werden Mock-Objekte eingesetzt? Allgemein und 3 Beispiele nennen.

Mocking dient nur Isolation von Komponenten, indem abhängige Objekte durch eine Art "Dummy" ersetzt werden. Mock-Objekte können dynamisch konfiguriert werden, sodass sie bei gegebener Eingabe (z.B. Aufruf einer Funktion) eine entsprechende Antwort liefern. Beispiele:

- Test von Service-Klassen durch Mocking vom darunter liegenden Data-Layer.
- Test einer Komponente, die auf ein externes System zugreifen muss, das möglicherweise langsam oder kostenpflichtig ist (für automatisierte, regelmäßig durchgeführte Tests ungeeignet).
- Als Ersatz für Komponenten, die nur schwer künstlich in einen gewissen Zustand gebracht werden können.

## Was bedeutet Refactoring und wann wird es eingesetzt? Wie wird Refactoring durchgeführt?

Refaktorisierung ist die Umgestaltung (= Umstrukturierung) von Programmcode, ohne die Programmfunktionalität dabei zu verändern, mit den Zielen, die Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit des Codes zu verbessern. Refaktorisierung wird auch zur Vermeidung von Redundanzen (z.B. mittels Ersetzung durch generischer Klassen) eingesetzt.