Einzelprüfung „Datenbanksysteme / Softwaretechnologie (vertieft)“

Einzelprüfungsnummer 66116 / 2013 / Herbst

# Thema 1 / Teilaufgabe 2 / Aufgabe 3 (*Vermischte Softwaresysteme-Fragen*)

**Stichwörter:** Testen, Model Checking, Refactoring, EXtreme Programming, White-Box-Testing, Black-Box-Testing, Funktionale Anforderungen, Nicht-funktionale Anforderungen, Kontinuierliche Integration (Continuous Integration), Unit-Test, wp-Kalkül

---

(a) Nennen Sie jeweils einen Vorteil und einen Nachteil für Qualitätssicherung durch *„Testing“* bzw. durch *„Model Checking“*.

Lösungsvorschlag

**Qualitätssicherung durch „Testing“**

**Vorteil** schnell, Massentests

**Nachteil** keine 100% Garantie, dass alles getestet ist

**Qualitätssicherung durch „Model Checking“**

**Vorteil** mathematischer Beweis

**Nachteil** teilweise langwierig / nicht möglich

(b) Definieren Sie den Begriff „Refactoring“.

Lösungsvorschlag

Verbesserung der Code-/Systemstruktur mit dem Ziel einer besseren Wartbarkeit

- Dokumentation, Namen, Kommentare
- keine neuen Funktionalitäten
- bessere Struktur, einheitlich, einfacher

(c) Begründen Sie, warum bei der Entwicklung nach der Methode des „eXtreme Programming“ langfristig gesehen Refactorings zwingend notwendig werden.

Lösungsvorschlag

Im „eXtreme Programming“ wird das Projekt kontinuierlich aufgebaut, somit ist ein Refactoring, auch aufgrund des Pair-Programmings, auf lange Sicht gesehen notwendig.

(d) Wie wird in der Praxis während und nach erfolgtem Refactoring sichergestellt, dass keine neuen Defekte eingeführt werden bzw. wurden?

Lösungsvorschlag

Re-testing → erneute Verifikation mit Tests nach Refactoring

(e) Worin besteht der Unterschied zwischen „White-Box-Testing“ und „Black-Box-Testing“?

Lösungsvorschlag

**White-Box-Testing:** Struktur-Test → Wie funktioniert der Code?

**Black-Box-Testing:** Funktionstest → Das nach außen sichtbare Verhalten wird getestet.

(f) Nennen Sie vier Qualitätsmerkmale von Software.

Lösungsvorschlag

Änderbarkeit, Wartbarkeit, gute Dokumentation, Effizienz, Funktionalität (→ Korrektheit), Zuverlässigkeit, Portabilität

(g) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Lösungsvorschlag

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

(h) Was verbirgt sich hinter dem Begriff „Continuous Integration” ?

Lösungsvorschlag

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (→ neue Programmversion).

(i) Nennen Sie sechs Herausstellungsmerkmale des *„eXtreme Programming“* Ansatzes.

Lösungsvorschlag

- Werte: Mut, Respekt, Einfachheit, Feedback, Kommunikation
- geringe Bedeutung von formalisiertem Vorgehen, Lösen der Programmieraufgabe im Vordergrund
- fortlaufende Iterationen
- Teamarbeit und Kommunikation (auch mit Kunden)
- Ziel: Software schneller und mit höherer Qualität bereitstellen, höhere Kundenzufriedenheit
- Continuous Integration und Testing, Prototyping
- Risikoanalysen zur Risikominimierung

- YAGNI (You ain't gonna need it) → nur die Features, die gefordert sind, umsetzen; kein Vielleicht braucht man's... "

(j) Was versteht man unter einem Unit-Test? Begründen Sie, warum es unzureichend ist, wenn eine Test-Suite ausschließlich Unit-Tests enthält.

Lösungsvorschlag

Unter einem Unit-Test versteht man den Test eines einzelnen Software-Moduls oder auch nur einer Methode. Dies ist allein nicht ausreichend, da man so nichts über das Zusammenspiel der Module aussagen kann.

(k) Nennen Sie jeweils eine Methodik, mit welcher in der Praxis die Prozesse der *„Validierung"* und der *„Verifikation"* durchgeführt werden.

Lösungsvorschlag

**Methodik der Verifikation:** Testen, wp-Kalkül, Model Checking

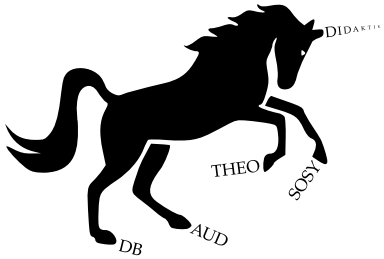
**Methodik der Validierung:** Kundentest, Kundengespräch (Spezifiaktion durchsprechen)

(l) Grenzen Sie die Begriffe *„Fault"* und *„Failure"* voneinander ab.

Lösungsvorschlag

**Fault:** interner Fehlerzustand, der nach außen nicht nicht sichtbar werden muss, aber kann.

**Failure:** Systemfehler / Fehlerzustand, der nach außen sichtbar wird.




**Die Bschlangaul-Sammlung**
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.




Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66116/2013/09/Thema-1/Teilaufgabe-2/Aufgabe-3.tex`