3. Testverfahren

Um die Qualität von Software garantieren zu können, genügt es nicht, erst am Schluss der Entwicklung die Software zu testen. Es ist wichtig, dass der Qualitätsprozess den ganzen Entwicklungszyklus begleitet. Je nach Phase (Analyse, Design, Implementation, Test) kommen dabei andere Verfahren zum Einsatz.

Inhalt

- Audit
- Review
- Black-Box Test
- White-Box Test

Audit

Bei einem Audit wird z.B. der Rahmen der Qualitätssicherung festgelegt und überwacht. Audit können auf einzelne Produkte als auch auf Vorgehensmodelle (SW-Entwicklungsprozesse) angewendet werden. Bei kleinen Projekten braucht es mitunter keine Audit, sofern ein gültiges Vorgehensmodell existiert. Bei grösseren Projekten können auch mehrere Audit stattfinden, sofern Umfang und Aufwand dies erfordern.

Review

Das Review dient der Kontrolle von beliebigen Dokumenten des SW-Entwicklungsprozesses. Dabei wird in einem Team das „Produkt“ begutachtet und entsprechend beurteilt. Dokumente für ein Review können sein:

- Prozessanalysen
- Funktionsbeschreibungen
- Struktogramme
- Programmcode
- usw.

Beim **Code Review** können unterschiedliche Aspekte beurteilt werden, wie

- Form des Programms, d.h. die Einhaltung von Coding Guidelines
- Dokumentation der Programmmodule
- Korrektheit des Programmcodes

Black-Box Test

Der Black-Box Test dient der Untersuchung des *Verhaltens* eines Programms. Dabei werden dem Programm Testdaten zugeführt. Diese Daten werden im Vorfeld – korrekterweise sogar bereits vor der Implementierung – festgelegt. Sie sind so zu wählen, dass der gewünschte Verarbeitungsvorgang

überprüft werden kann. Es geht hier um die Frage, WAS das Programm macht. Die Qualität des Tests hängt ausschliesslich von den Testdaten ab. Es ist daher wichtig, dass alle möglichen Kombinationen von Daten festgelegt und somit auch getestet werden.

> Beispiel: Bei einem Taschenrechner müssen alle Operationen zwingend auch mit negativen Zahlen und – ganz besonders – der Zahl 0 getestete werden.

Der Black Box Test stellt immer nur eine Stichprobe dar und kann keine Garantie dafür abgeben, dass die Anwendung in allen Situationen richtig läuft. Es genügt ein Spezialfall, der nicht getestet wurde, um Probleme zu haben.

Äquivalenzklassen und Grenzwertanalyse helfen dabei, um geeigneten **Testfälle** zu finden. Dabei wird der Wertebereich in Abschnitte unterteilt durch Grenzwerte (z.B. Eintritt ab 18). Ein Wert aus einem solchen Abschnitt (z.B. alle Zahlen alle ab 18) kann stellvertretend für den ganzen Bereich verwendet werden. Dies ergibt einen Testfall pro Abschnitt sowie drei Testfälle pro Grenzwert (d.h. für die Grenze und ein Wert davor und danach, also 17, 18, 19). Weitere Infos finden Sie hier.

Um den Black-Box Test systematisch durchzuführen, sind die nötigen Testfälle im Voraus festzulegen. Zu einem Testfall gehören 1. die **Testdaten** sowie 2. das **erwartete Verhalten** und 3. das **tatsächliche Verhalten** der Anwendung. Sofern Soll und Ist übereinstimmen, ist der Testfall korrekt abgelaufen. Wenn nicht, ist eine Anpassung der Software nötig.

White-Box Test

Der White-Box Test dient der Untersuchung des Programmcodes. Dabei kann unterschieden werden in korrekten Ablauf und korrekte Datenwerte. Im Vordergrund steht hier, WIE das Programm abläuft. Wie sieht der Inhalt einer Variablen aus? Welche Pfade werden durchlaufen?

Das Ziel einer vollen **Pfadabdeckung** (resp. Code Coverage) ist es, dass jede Anweisung und somit jeder Programmpfad mindestens einmal durchlaufen und somit die funktional korrekte Ausführung bewiesen wird. Mit diesem Test werden auch die Bedingungen in `if`, `while` und `for` Statements geprüft, damit ersichtlich ist, wann ein bestimmter Pfad nicht durchlaufen wird. In einer modernen IDE kann die Code Coverage i.d.R. leicht angezeigt werden.

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu90/theorie/lu4-kapitel_3

Last update: **2026/08/30 14:06**