Software-Qualität

Das Wort Qualität steht für viele verschiedene Aspekte eines Produktes oder einer Dienstleistung. Täglich beschäftigen wir uns mit Qualität, auch wenn wir dieses Wort nicht dazu gebrauchen.

Inhaltsverzeichnis

1. Auswirkungen von Fehlern
2. Arten von Fehlern
3. Normierte Qualitätsmerkmale
4. Testaktivitäten und -planung
5. Dynamische Testmethoden
    1. White-Box-Testmethode
    2. Black-Box-Testmethode

Auswirkungen von Fehlern

In der Praxis können kleine Fehler grosse Konsequenzen haben:

- Prestigeverlust
- frustrierte User/Kunden, usw.
- Programmabbrüche
- Datenverfälschung - Anomalien
- falsches Systemverhalten
- Systemabstürze (betrifft auch andere Programme und kommt dem GAU sehr nahe)

Die Folgen und Kosten von Fehlern hängen stark von der Art des Einsatzes eines Produktes ab:

- Massenprodukt (z.B. internationaler Webauftritt mit einer grossen Anwender-Anzahl, die dem Hersteller nicht direkt bekannt sind)
- individuelle Software (kleiner Anwenderkreis mit Integration in den Entwicklungsprozess)

Arten von Fehlern

Welche Fehlerarten gibt es? Nachfolgend eine Auswahl:

- Codierfehler (z.B. Syntax, undefinierte Variablen)
- Logikfehler (z.B. bei Entscheidungen, Endlos-Schleifen)
- Entwurfsfehler (z.B. Fehler im Konzept)
- Datenfehler (z.B. Mathematische Operationen mit Strings statt mit Zahlen)
- Umsetzungsfehler (von Spezifikation ins Programm)
- Oberflächenfehler (bei GUI oder sonstiger Benutzerinteraktion)

Normierte Qualitätsmerkmale

Was kann der Kunde erwarten, wenn er Ihnen eine Software (z.B. Webblog oder ein CMS inkl. Styling, JavaScript, PHP und Datenbank) in Auftrag gibt?



Abb-1: Erwartungen des Kunden an der Software-Qualität

Es gibt eine Norm (DIN/ISO 9126), welche die wichtigsten Qualitätsmerkmale regelt:

1. **Zuverlässigkeit (reliability)**: Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren.
2. **Funktionalität (functionality)**: Vorhandensein von Funktionen mit festgelegten Eigenschaften. Diese Funktionen erfüllen die definierten Anforderungen.
3. **Benutzbarkeit (usability)**: Der Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe.
4. **Effizienz (efficiency)**: Das Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen.
5. **Änderbarkeit (maintainability)**: Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen und der funktionalen Spezifikationen einschließen.
6. **Übertragbarkeit (portability)**: Eignung der Software, von einer Umgebung in eine andere übertragen zu werden. Umgebung kann organisatorische Umgebung, Hardware- oder Software-Umgebung einschließen.

Testaktivitäten und -planung

Die Testaktivitäten können nicht kurz vor Schluss durchgeführt werden. Das Testen von Software muss geplant sein, nimmt es doch bei grossen Projekten ca. 15%–30% in Anspruch.

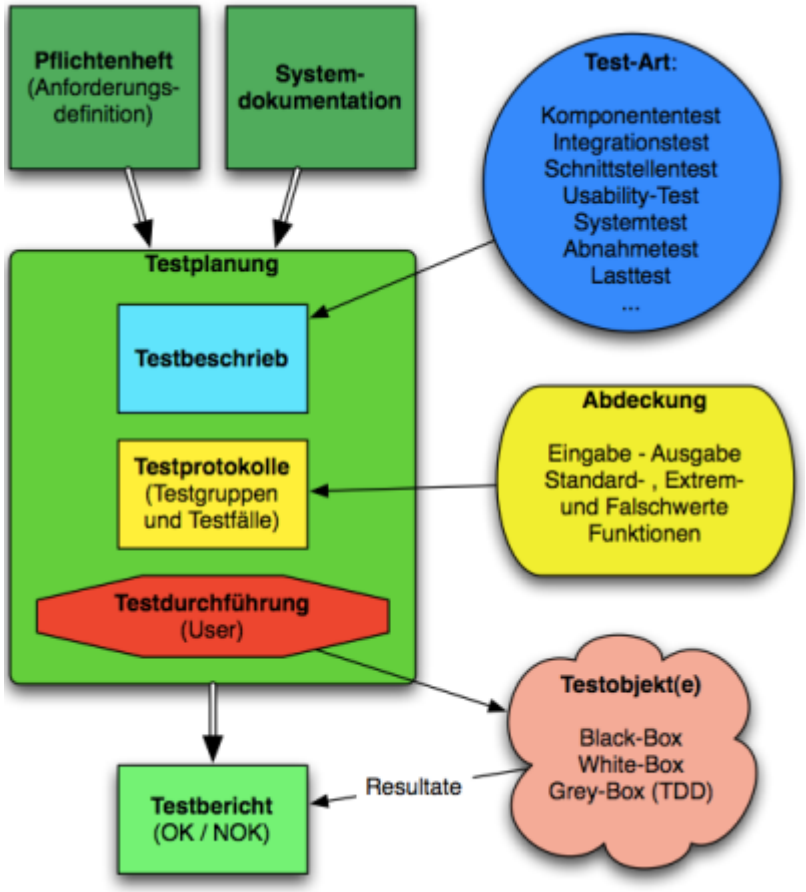


Abb-2: Testaktivitäten und -planung

Es gibt für das Durchführen von Testaktivitäten einige Prinzipien, die sich in der Praxis bewährt haben:

- **Fehler möglichst frühzeitig aufdecken**: Das heisst, ein Test sollte frühzeitig und entwicklungsbegleitend erfolgen, um Fehlerauswirkungen und Folgefehler zu minimieren.
- **Testziele erreichbar und messbar formulieren**: Das heisst, dass überprüfbare Ziele (z.B. Forderungen über auszuführende Testfälle) für den Test formuliert werden sollten. Das Testende ist erreicht, wenn die zuvor aufgestellten Testziele erreicht werden.
- **Testfälle verwalten**: Das heisst, die Testfälle sind zu speichern und für spätere Testwiederholungen (Regressionstests) verfügbar zu machen.
- **Testaktivitäten planen**: Das heisst, ohne Planung werden Testaktivitäten unsystematisch und im Wortsinne „ungeplant“ durchgeführt. Die Testplanung ist in die Projektplanung zu integrieren.
- **Testaktivitäten dokumentieren**: Das heisst, nur durch Dokumentation ist Transparenz und Revisionssicherheit der Tests gewährleistet. Zur Testdokumentation gehören Testpläne, Testspezifikationen, Testfallbeschreibungen, sowie Test- und Fehlerprotokolle.

Dynamische Testmethoden

Bei den dynamischen Testmethoden wird die Software gestartet und nach bestimmten Testkriterien durchlaufen. Es gibt einige Grundsätze für die Testfallermittlung. Details zu Testfällen und wie diese erstellt werden finden Sie unter Testbericht und Testprotokoll erstellen.

White-Box-Testmethode

Bei der White-Box-Methode werden die Testfälle mit Kenntnis der internen Strukturen des Testobjekts (Source-Code) entwickelt, d.h. diese Methode kann nur von Entwicklern eingesetzt werden.

Der laufende Test beim Programmieren ist ebenfalls ein White-Box-Test. Die meisten Entwicklungsumgebungen stellen dazu umfangreiche Tools zur Verfügung (DEBUG), es sind aber auch eigenständige Applikationen für White-Box-Tests auf dem Markt erhältlich.

Black-Box-Testmethode

Bei dieser Methode werden die Testfälle aus der vorliegenden Spezifikation (z.B. Pflichtenheft) oder aber aus der Oberflächenstruktur (z.B. Erfassungsmasken, GUI) hergeleitet. Die innere Struktur des Objektes wird nicht berücksichtigt und kann unbekannt sein. Diese Testmethode kann also auch vom Anwender durchgeführt werden.

Diese Methoden können sinnvollerweise auch kombiniert werden.

Quellen: TBZ-Modul 103: Testen und Dokumentieren

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

Permanent link:
https://wiki.bzz.ch/modul/m306/articles/05?rev=1776173104

Last update: **2026/04/14 15:25**