LU07a - Einführung Software-Qualität

Auswirkungen von Fehlern

In der Praxis können kleine Fehler grosse Konsequenzen haben:

- Prestigeverlust
- frustrierte User/Kunden, usw.
- Programmabbrüche
- Datenverfälschung - Anomalien
- falsches Systemverhalten
- Systemabstürze

Die Folgen und Kosten von Fehlern hängen stark vom Einsatzgebiet der Software ab – hier zwei Beispiele:

Massenprodukt (z.B. internationaler Webauftritt)

- Sehr viele Benutzer*innen betroffen
- Support- und Schulungskosten steigen
- Reputationsschäden durch negative Nutzererfahrung

Individuelle Software (z.B. firmeninterne Anwendung):

- Kleiner, oft bekannter Nutzerkreis
- Verbesserungen können schneller integriert werden
- Fehler wirken sich punktuell aus, aber können Arbeitsprozesse stören

Arten von Fehlern

Welche Fehlerarten gibt es? Nachfolgend eine Auswahl:

- Codierfehler (z.B. Syntax, undefinierte Variablen)
- Logikfehler (z.B. falsche Bedingungen bei Entscheidungen, Endlos-Schleifen)
- Entwurfsfehler (z.B. konzeptionelle Schwächen oder falsche/fehlende Anforderungen)
- Datenfehler (z.B. mathematische Operationen mit Strings statt mit Zahlen)
- Umsetzungsfehler (Fehler bei der Übersetzung von Spezifikationen in Code)
- Oberflächenfehler (schwer verständliche Benutzeroberflächen, fehlende Rückmeldung bei Interaktionen, etc.)

Usability-Fehler können in sicherheitskritischen Bereichen wie Medizin, Luftfahrt oder autonomen Fahrzeugen dramatische Auswirkungen haben.

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?

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

- **Zuverlässigkeit (reliability):** Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren.
- **Funktionalität (functionality):** Vorhandensein von Funktionen mit festgelegten Eigenschaften. Diese Funktionen erfüllen die definierten Anforderungen.
- **Benutzbarkeit (usability):** Der Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe.
- **Effizienz (efficiency):** Das Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen.
- **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.
- **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.

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:** Überprüfbare Ziele (z.B. Forderungen über auszuführende Testfälle) sollten formuliert werden. Das Testende ist erreicht, wenn die zuvor aufgestellten Testziele erfüllt sind.
- **Testfälle verwalten:** Testfälle sind zu speichern und für spätere Testwiederholungen (Regressionstests) verfügbar zu machen.
- **Testaktivitäten planen:** Ohne Planung werden Testaktivitäten unsystematisch und „ungeplant“ durchgeführt. Die Testplanung ist in die Projektplanung zu integrieren.
- **Testaktivitäten dokumentieren:** Nur durch Dokumentation ist Transparenz und Revisionssicherheit gewährleistet. Zur Testdokumentation gehören Testpläne, Testspezifikationen, Testfallbeschreibungen sowie Test- und Fehlerprotokolle.

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.