Software Engineering I

7. Test und Qualitätssicherung

Prof. Dr. Eckhard Kruse

DHBW Mannheim

Softwarequalität

Definition

Softwarequalität ist die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen.

(DIN ISO 9126)

- Spezifikation der gewünschten Eigenschaften ist Voraussetzung, um die Qualität zu beurteilen.
- Wichtige Software-Qualitätsmerkmale:
  - Korrektheit: Software entspricht den Anforderungen
  - Zuverlässigkeit: Geringe Ausfallzeiten/Ausfallwahrscheinlichkeit
  - Robustheit: Unempfindlich gegenüber (kleineren) Fehlern/Störungen
- Qualität ist schwer messbar (im Gegensatz zu Hardware, z.B.: Fertigungstoleranzen, statistische Ausfallwahrscheinlichkeiten, Verfügbarkeit)
- Maßnahmen zur Qualitätssteigerung, z.B.:
  - Testen (und ggf. formale Verifikation)
  - Audits und Reviews
  - Definierte Entwicklungsprozesse

Qualitätssicherung im V-Modell

QS-Handbuch

- Kurzbeschreibung der Qualitätsziele im Projekt
- Festlegung zu prüfender Produkte und Prozesse
- Organisation und Vorgaben für die Planung und Durchführung der Qualitätssicherung im Projekt (und von externen Zulieferungen.)

QS-Verantwortlicher

Grundlage für

QS-Bericht

- Zentrales Dokument zur Beurteilung der Produktqualität
- Beschreibt Umfang der durchgeführten Prüfungen, aufgetretene Qualitätsprobleme und Maßnahmen zur Behebung
- Zur projektinternen und externen Berichterstattung

Input

Prüfspezifikationen

- Vorgabe und Anleitung für die Prüfung
- für Produktversionen und Teilprodukte

definiert

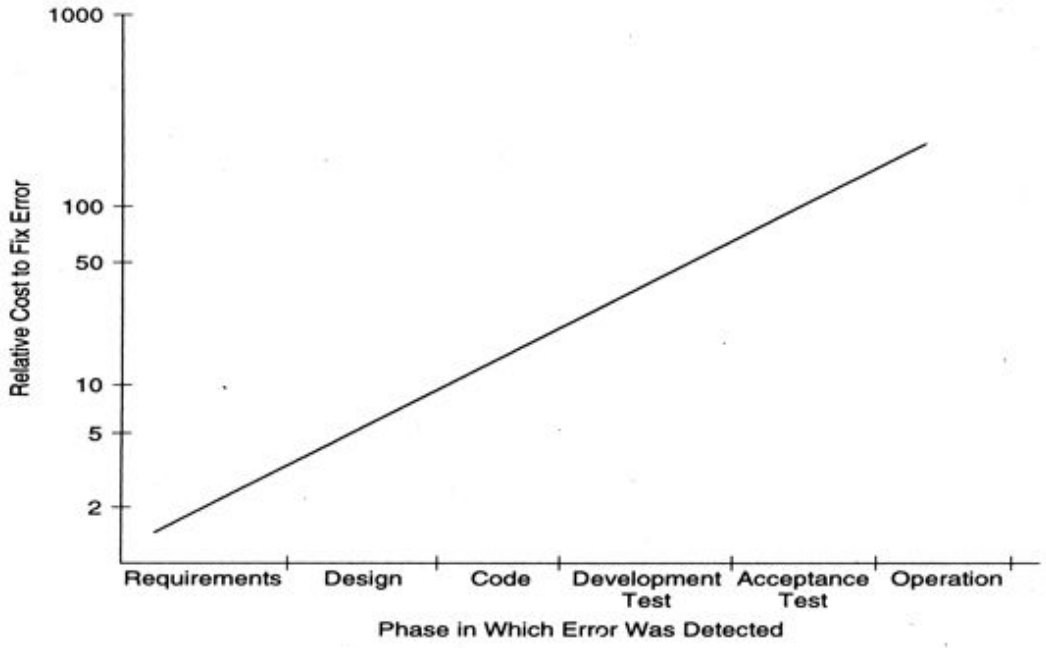
Prüfprotokolle

- Aufzeichnungen über den Verlauf der Prüfung,
- Gegenüberstellung Ist- und Soll-Ergebnisse
- Analyse, ggf. Lösungsvorschläge
- Wichtig: Reproduzierbarkeit

Prüfer

Kosten von Fehlern

- Fehler können in jeder Phase des Entwicklungsprozesses auftreten.
- Je größer die Zeit zwischen Ursache und Behebung des Fehlers, desto größer die Kosten
- Qualitätssicherung ist von Anfang an wichtig!



Verifikation und Validierung

Definition

Verifikation **(Verification)** ist die Prüfung, dass ein Softwareartefakt die korrekten, d.h. spezifizierten Ergebnisse liefert.

Are we doing the product right?

- Modultests, Komponententests, Integrationstests
- Reviews mit Entwicklern
- Formale Verifikation: Mathematische Beweisverfahren, um die Korrektheit von Software (meist nur von kleinen Teilsystemen) zu beweisen.

Validierung **(Validation)** ist die Prüfung der Eignung einer Software bezogen auf ihren Einsatzzweck.

Are we doing the right product?

- Abnahmetest, Akzeptanztest (user acceptance test): Werden aus Sicht des Kunden/Anwender mit der Software die angestrebten Ziele erreicht?
- Reviews mit Kunden/Anwendern

Nicht-funktionale Eigenschaften

Die Softwarequalität bzgl. nicht-funktionaler Eigenschaften lässt sich oft nur sehr schwierig nachweisen/quantifizieren.

Beispiel **Zuverlässigkeit**:

- Wahrscheinlichkeit, dass sich ein System in einem gegebenen Zeitraum erwartungsgemäß verhält
- Hardware-Zuverlässigkeitsmaße, z.B.
  - Mean time between failure (MTBF)
  - Mean time to repair (MTTR)
  - z.B. Hard disk: MTBF = 500000 h
  - Und wenn das nicht reicht → Redundanz

Und Software-Zuverlässigkeit?
MTBF, Redundanz, ...?

Fault, Error, Failure

Definition

Fault/Defect/Bug:
Fehlerhafte Stelle im System, die ein Fehlverhalten auslösen kann.

Error
Diskrepanz zwischen einem tatsächlichen Wert/Verhalten im System und dem gewünschten, theoretischem Wert/Verhalten.

Failure
Tatsächlich auftretendes, fehlerhaftes Verhalten des Systems, d.h. das Programm ist zu einem bestimmten Zeitpunkt nicht mehr in der Lage, seine korrekte Funktion zu erledigen.

Damit **Faults** nicht automatisch zu **Failures** führen:

- Fault tolerance
- Error handling / Exception handling
- Redundancy

Testspezifikation

Die Testspezifikation beschreibt die durchzuführenden Tests und die erwarteten Ergebnisse.

- Definition von verschiedenen Testfällen:
  - oft in direkter Anlehnung an die Use Cases, d.h. typische Abläufe aus Anwendersicht
- Für jeden Testfall: Schritt-für-Schritt-Auflistung der Vorbedingungen, Eingabe und des erwarteten (überprüfbaren) Ergebnisses.
  - *"Anwender klickt auf Menüpunkt xy"*
  - *"Neues Fenster wird eingeblendet, Daten werden gespeichert usw."*
- Wichtig: Auch Sonderfälle und unerlaubte Eingaben sollten getestet werden.
- Typisch: Black-Box-Testing = Testen aus Sicht der Anwenders, der kein Wissen über den inneren Aufbau des Systems hat.

Regressionstests

Definition

Unter **Regressionstests** (regression tests) versteht man die regelmäßige Wiederholung von identischen Testfällen, um auch in bereits getesteten Softwareteilen eventuell nachträglich entstandene Fehler (z.B. aufgrund von Nebenwirkungen von Änderungen) zu erkennen.

- Aufgrund der regelmäßigen Wiederholung (und des damit verbundenen Aufwands) sollten Regressionstest möglichst automatisiert erfolgen.
- Sinnvoller Ansatz z.B.: Automatisiertes Durchführen von Regressionstests im Rahmen des Buildprozesses: Jedes erfolgreiche Build wird automatisch einer Reihe von Systemtests unterzogen.

Testabdeckung

Definition

Unter **Testabdeckung** (test coverage / code coverage) versteht man den prozentualen Anteil von getesten Softwareteilen im Verhältnis zum Gesamtumfang der Software.

- Meist nur ein geschätztes, theoretisches Maß
- Die Abdeckung kann sich auf verschiedene Kriterien beziehen (mit zunehmendem Aufwand):
  - Funktionen
  - Befehle
  - Bedingungen
  - Ausführungspfade
- 100%ige Testabdeckung lässt sich nur sehr schwer garantieren (komplexe analytische Verfahren, formale Methoden)

Beispiele

Beispiel

- Quality Plan – Document template
- Test plan – template
- Testspezifikation: Einfaches Template

Lieferant: Code Freeze + Testen

Software-Engineering-Projekt

P.17 Lieferant: Code Freeze+Testen

Die Implementierung aller Funktionen des Softwaresystems ist abgeschlossen. Die verbleibenden Entwicklungsaktivitäten sollen sich auf das Testen und Beheben von Fehlern konzentrieren. Um nicht in der letzten Projektphase noch neue Risiken und Instabilität in das System hineinzubringen, wird typischerweise ein „Code Freeze“, d.h. ein Einfrieren des Codes beschlossen. Erforderliche Änderungen der Implementierung (um Fehler zu beheben) sind nur noch nach Absprache und Abwägen möglicher Risiken möglich. Überlegen Sie, wie Sie Ihr System systematisch testen wollen.

a) Wie dokumentieren Sie erkannte Fehler und den Status der Fehlerbehebung (Schwere des Fehlers, soll er behoben werden, Prioritäten usw.).

b) Bereiten Sie sich vor (ggf. mit ein, zwei Folien), dem Qualitätsmanager zu berichten, wie Sie in der Testphase obige Schritte umsetzen, um eine hohe Softwarequalität sicherzustellen.

Kunde-Lieferant: Übergabe + Abnahme

Software-Engineering-Projekt

P.18 Kunde-Lieferant: Übergabe und Endabnahme

Der Lieferant hat einen Termin beim Kunden, um das Produkt zu übergeben und den Kunden einzuweisen. Der Kunde macht eine Endabnahme. Ist alles in Ordnung, so dass die endgültige Rechnungsstellung und Bezahlung erfolgen können?

Lieferant: Interner Abschluss

Software-Engineering-Projekt

P.19 Lieferant: Interne Abschlusspräsentation

Ihr Management möchte in einem Abschlussvortrag über Ihr Projekt informiert werden. War das Projekt ein Erfolg, konnten die Ergebnisse wie gewünscht an den Kunden ausgeliefert werden und ist der Kunde zufrieden? Lessons learnt: Was lief gut, was hätte rückblickend anders gemacht werden sollen? Können Sie das Management überzeugen, dass Sie sich mit diesem Projekt für weitere Projektleitungsaufgaben qualifizieren?