

Triona - Information und Technologie GmbH

Basiswissen Software-Test und -Qualität komprimiert

Marc Brosius, 05.September 2008

Software-Test

Inhalte (keine Reihenfolge)

- Warum? Grundlagen, Motivation und Begriffsbildung
- Wann? Die Rolle des Tests im Softwarelebenszyklus
- Was und wie? Testobjekte, Testarten und -methoden
- Womit? Werkzeuge und Frameworks

Grundlagen und Begriffsbildung

Test, was ist das?

„Ein Test (aus dem altfranzösischen: test Tiegel, Topf für alchemistische Versuche) ist ein Versuch, mit dem größere Sicherheit darüber gewonnen werden soll, ob ein technischer Apparat oder ein Vorgang innerhalb der geplanten Rahmenbedingungen funktioniert oder nicht bzw. ob bestimmte Eigenschaften vorliegen oder nicht. ...“

„...Bei dem Softwaretest bezeichnet Test die Ausführung eines Programms auf einem Computer zum Aufspüren von Programmfehlern (dynamischer Test) und den statischen Test bei dem der Code analysiert wird. ...“

([URL: WIKI] Wikipedia zum Thema „Test“)

“Der Softwaretest ist ein Test, der während der Softwareentwicklung durchgeführt wird. Als analytische Maßnahme zur Qualitätssicherung soll er sicherstellen, dass eine Software die an sie gestellten Anforderungen in der geforderten Qualität erfüllt.“

([URL: WIKI] Wikipedia zum Thema „Softwaretest“)

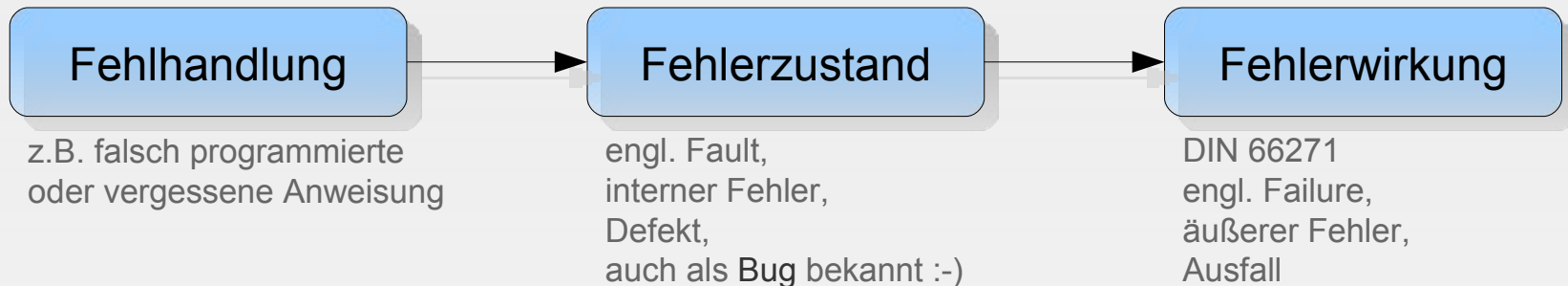
Grundlagen und Begriffsbildung

Der Fehlerbegriff

während Test bzw. Betrieb festgelegte Anforderung bzw. Spezifikation

Ausgangspunkt: erwartetes, korrektes, also nicht fehlerbehaftetes Verhalten

- Ein Fehler ist eine Abweichung zwischen Ist-Verhalten und Soll-Verhalten. Ein Mangel liegt vor, wenn eine Anforderung oder berechnete Erwartung nicht angemessen erfüllt wird. (abgeschwächte Form eines Fehlers)
- Ein Fehler ist schon seit der Entwicklung in der Software, kommt jedoch erst mit ihrer Ausführung zum Tragen. (Software altert nicht)





Grundlagen und Begriffsbildung

Der Testbegriff

„Testen ist nicht Debugging“ [Spillner 07]

Debugging

Das Lokalisieren und Beheben eines Defekts/Bugs
Aufgabe des Entwicklers
Führt zu einer Änderung

Testen

Das gezielte und systematische Aufdecken von Fehlerwirkungen
Ausführung des Testobjekts zur Überprüfung



Grundlagen und Begriffsbildung

Ziele des Software-Tests

- Motivation:
Kein umfangreiches Softwaresystem ist fehlerfrei.
- Ziele:
 - Ausführung des Programms mit dem Ziel, Fehlerwirkungen nachzuweisen
 - Ausführung des Programms mit dem Ziel die Qualität zu bestimmen
 - Ausführung des Programms mit dem Ziel das Vertrauen in das Programm zu erhöhen
 - Analysieren des Programms oder der Dokumentation um Fehlerwirkungen vorzubeugen
- Aber:
Komplexität → Ausnahmesituationen werden nicht bedacht (in Entwicklung und Test)
→ Fehlerfreiheit durch Testen nicht erreichbar oder nachweisbar



Grundlagen und Begriffsbildung

Software-Qualität

Merkmale nach ISO 9126:

- Funktionalität
- Zuverlässigkeit
- Benutzbarkeit
- Effizienz
- Änderbarkeit
- Übertragbarkeit



Grundlagen und Begriffsbildung

Software-Qualität

Merkmale nach ISO 9126:

Funktionalität

Vorhandensein von Funktionen mit festgelegten Eigenschaften. Diese Funktionen erfüllen die definierten Anforderungen.

| | |
|--------------------|---|
| Angemessenheit: | Eignung von Funktionen für spezifizierte Aufgaben |
| Richtigkeit: | Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen |
| Interoperabilität: | Fähigkeit, mit vorgegebenen Systemen zusammenzuwirken |
| Sicherheit: | Fähigkeit, unberechtigten Zugriff, sowohl versehentlich als auch vorsätzlich, auf Programme und Daten zu verhindern |
| Konformität: | Grad, in dem die Software Normen oder Vereinbarungen zur Funktionalität erfüllt. |



Grundlagen und Begriffsbildung

Software-Qualität

Merkmale nach ISO 9126:

Zuverlässigkeit

Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren.

| | |
|------------------------|--|
| Reife: | Geringe Versagenshäufigkeit durch Fehlerzustände. |
| Fehlertoleranz: | Fähigkeit, ein spezifiziertes Leistungsniveau bei Software-Fehlern oder Nicht-Einhaltung ihrer spezifizierten Schnittstelle zu bewahren. |
| Robustheit: | Fähigkeit, ein stabiles System bei Eingaben zu gewährleisten, die gar nicht vorgesehen sind. |
| Wiederherstellbarkeit: | Fähigkeit, bei einem Versagen das Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen. |
| Konformität: | Grad, in dem die Software Normen oder Vereinbarungen zur Zuverlässigkeit erfüllt. |



Grundlagen und Begriffsbildung

Software-Qualität

Merkmale nach ISO 9126:

Benutzbarkeit

Aufwand, der zur Benutzung erforderlich ist, und individuelle Beurteilung der Benutzung durch eine festgelegte oder vorausgesetzte Benutzergruppe.

| | |
|-------------------|--|
| Verständlichkeit: | Aufwand für den Benutzer, das Konzept und die Anwendung zu verstehen. |
| Erlernbarkeit: | Aufwand für den Benutzer, die Anwendung zu erlernen (z.B. Bedienung, Ein-, Ausgabe). |
| Bedienbarkeit: | Aufwand für den Benutzer, die Anwendung zu bedienen. |
| Attraktivität: | Anziehungskraft der Anwendung gegenüber dem Benutzer. |
| Konformität: | Grad, in dem die Software Normen oder Vereinbarungen zur Benutzbarkeit erfüllt. |



Grundlagen und Begriffsbildung

Software-Qualität

Merkmale nach ISO 9126:

Effizienz

Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen.

- | | |
|----------------------|---|
| Zeitverhalten: | Antwort- und Verarbeitungszeiten sowie Durchsatz bei der Funktionsausführung. |
| Verbrauchsverhalten: | Anzahl und Dauer der benötigten Betriebsmittel bei der Erfüllung der Funktionen. Ressourcenbedarf, wie CPU-Zeit, Festplattenzugriffe usw. |
| Konformität: | Grad, in dem die Software Normen oder Vereinbarungen zur Effizienz erfüllt. |



Grundlagen und Begriffsbildung

Software-Qualität

Merkmale nach ISO 9126:

Änderbarkeit

Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist. Änderungen können Korrekturen, Verbesserungen oder Anpassungen an Änderungen der Umgebung, der Anforderungen oder der funktionalen Spezifikationen einschließen.

| | |
|--------------------|--|
| Analysierbarkeit: | Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen. |
| Modifizierbarkeit: | Aufwand zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder Anpassung an Umgebungsänderungen. |
| Stabilität: | Wahrscheinlichkeit des Auftretens unerwarteter Wirkungen von Änderungen. |
| Prüfbarkeit: | Aufwand, der zur Prüfung der geänderten Software notwendig ist. |



Grundlagen und Begriffsbildung

Software-Qualität

Merkmale nach ISO 9126:

Übertragbarkeit

Eignung der Software, von der Umgebung in eine andere übertragen werden zu können. Umgebung kann organisatorische Umgebung, Hardware- oder Software-Umgebung sein.

| | |
|--------------------|--|
| Anpassbarkeit: | Fähigkeit der Software, diese an verschiedene Umgebungen anzupassen. |
| Installierbarkeit: | Aufwand, der zum Installieren der Software in einer festgelegten Umgebung notwendig ist. |
| Koexistenz: | Fähigkeit der Software neben einer anderen mit ähnlichen oder gleichen Funktionen zu arbeiten. |
| Austauschbarkeit: | Möglichkeit, diese Software anstelle einer spezifizierten anderen in der Umgebung jener Software zu verwenden, sowie der dafür notwendige Aufwand. |
| Konformität: | Grad, in dem die Software Normen oder Vereinbarungen zur Übertragbarkeit erfüllt. |

Grundlagen und Begriffsbildung

Testaufwand

Testen kann Fehlerfreiheit nicht nachweisen.

Dazu wäre ein vollständiger Test nötig, der praktisch nicht durchführbar ist.

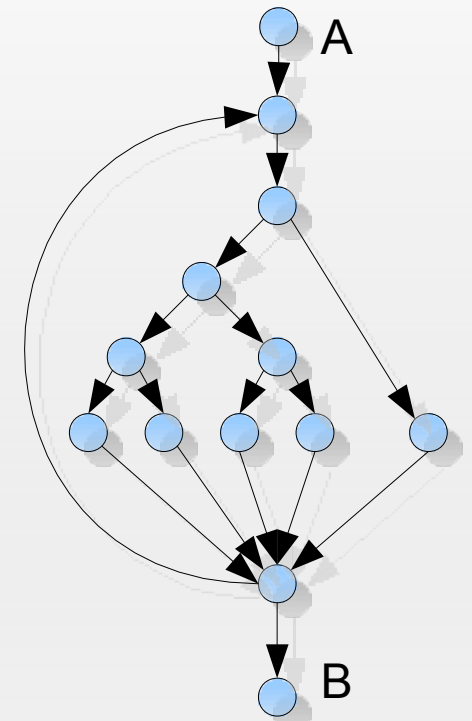
Grund: Vielzahl kombinatorischer Möglichkeiten
→ Anzahl Tests nahezu unbegrenzt

Bsp.: - 5 Pfade
- 20 Schleifendurchläufe

→ $5^{20} + 5^{19} + 5^{18} + \dots + 5^1 \approx 100 \cdot 10^{15}$ **100 Billionen !!!**

Manuell (à 5 min.): ca. 1 Milliarde Jahre

Automatisiert (à 5 μ s): ca. 19 Jahre



Grundlagen und Begriffsbildung

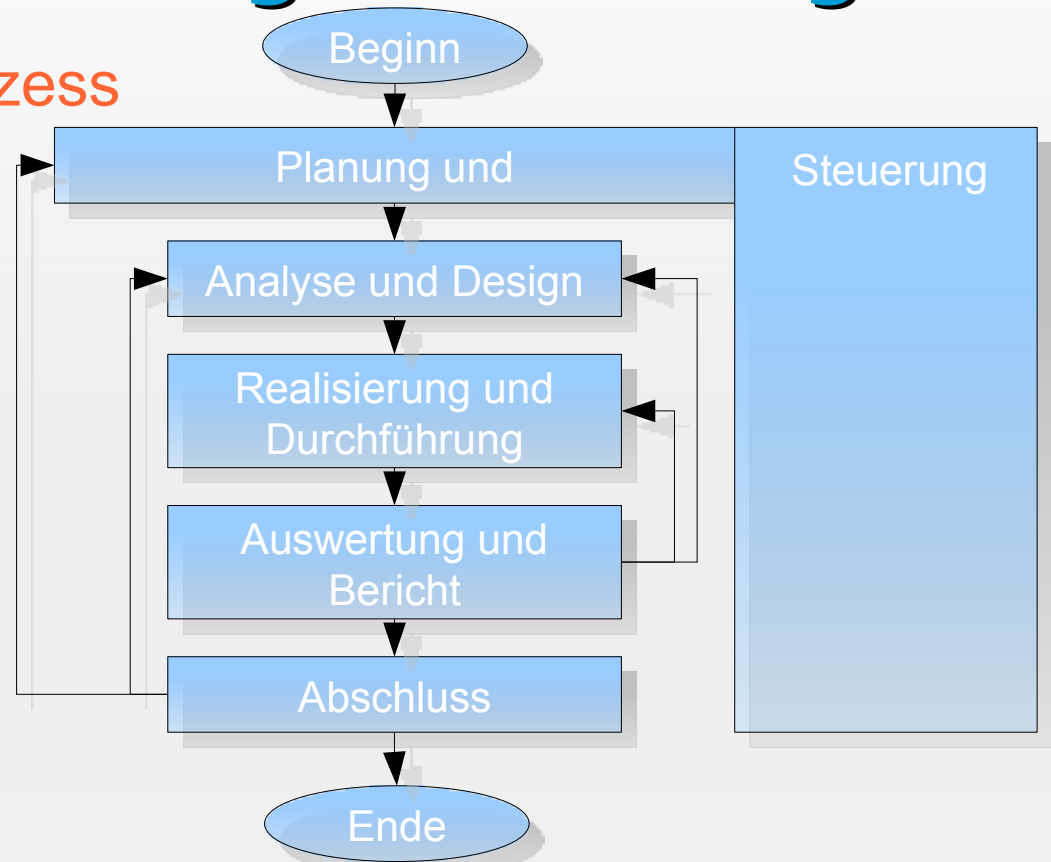
Weitere Begriffe

- Testprozess: Planung – Durchführung – Auswertung
- Testfall: festgelegte Randbedingungen
- Testszenario: Aneinanderreihung mehrerer Testfälle

Grundlagen und Begriffsbildung

Fundamentaler Testprozess

(in Anlehnung an den ISTQB-Lehrplan)



Grundlagen und Begriffsbildung

Testplanung und Steuerung

- Planung der Ressourcen
- Festlegung der Teststrategie
- Testintensität für Teilsysteme und einzelne Aspekte festlegen
- Priorisierung der Tests
- Werkzeugunterstützung

Grundlagen und Begriffsbildung

Testanalyse und Testdesign

- Logische und konkrete Testfälle
- Testfälle umfassen mehr als nur die Testdaten
- Testorakel
- Testfälle für erwartete und unerwartete Eingaben

Grundlagen und Begriffsbildung

Testrealisierung und Testdurchführung

- Ablauf der Testfälle
- Testrahmen
- Prüfung auf Vollständigkeit
- Prüfung der Hauptfunktionen
- Tests ohne Protokollierung sind wertlos
- Nachvollziehbarkeit und Reproduzierbarkeit sind wichtig
- Fehlerwirkung gefunden?
- Korrektur kann zu neuen Fehlerzuständen führen
- Die wichtigen Testfälle zuerst

Grundlagen und Begriffsbildung

Testauswertung und Bericht

- Testende erreicht?
- Weiterer Aufwand gerechtfertigt?
- Unerreichbarer Programmcode
- Weitere Kriterien für die Bestimmung des Testendes
- Mehrere Testzyklen berücksichtigen
- Endekriterien der Praxis: Zeit und Kosten
- Erfolgreiches Testen spart Kosten ein!

→ Testbericht



Grundlagen und Begriffsbildung

Grundsätze des Testens

1 Testen zeigt die Anwesenheit von Fehlern

Es ist kein Beweis für Fehlerfreiheit, auch wenn keine Fehlerwirkungen im Test gefunden wurden. Ausreichendes Testen verringert jedoch die Wahrscheinlichkeit, dass noch unentdeckte Fehlerzustände vorhanden sind.

2 Vollständiges Testen ist nicht möglich

Tests sind immer nur Stichproben und der Testaufwand muss deshalb nach Risiko und Prioritäten gesteuert werden.

3 Mit dem Testen frühzeitig beginnen

4 Häufung von Fehlern

Fehler treten nicht gleichverteilt im Testobjekt auf. Oft findet man mehrere Fehler, wo eine Fehlerwirkung gefunden wurden. Auf diesen Umstand muss flexibel reagiert werden.

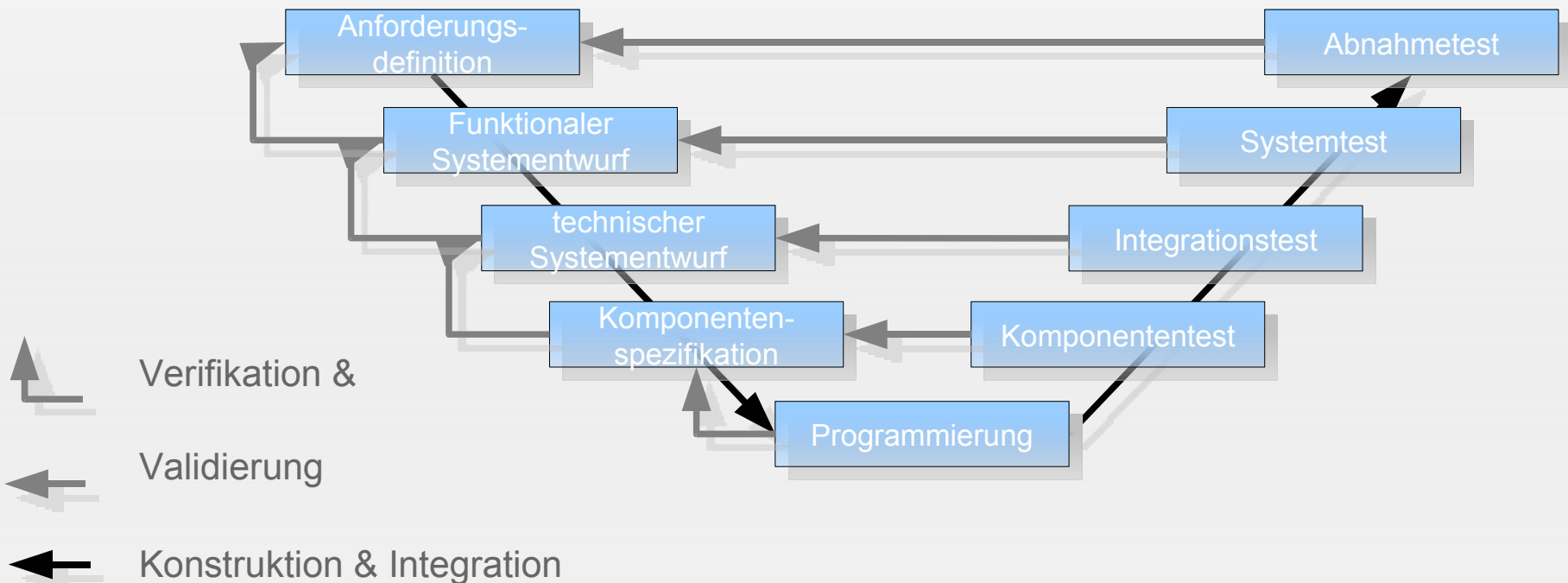
Grundlagen und Begriffsbildung

Grundsätze des Testens

- 5 Zunehmende Testresistenz**
Testfälle regelmäßig prüfen und durch neue oder veränderte ersetzen.
- 6 Testen ist abhängig vom Umfeld**
Datenschutzkritische Software verlangt andere Prüfungen als Spielesoftware.
- 7 Trugschluss: Keine Fehler bedeutet ein brauchbares System**
Akzeptanz der Nutzer

Die Rolle des Tests im Entwicklungsprozess

Beispiel: V-Modell





Die Rolle des Tests im Entwicklungsprozess

Komponententest

- Niedrigste Teststufe
- Testobjekte sind Module, in der OO-SWE meist Klassen, bei SOA meist einzelne Services
- Häufig vom Entwickler durchgeführt
- Bei agilen Prozessen wie XP:
 - Buddy-Testing. Gegenseitiger Test unter Kollegen
 - Test-First-Ansatz (test driven development)
- Da sehr entwicklungsnahe meist White-Box-Test (dazu später mehr)
 - Es werden i.d.R. Testtreiber benötigt
- Ziel: Prüfung auf korrekte Funktionalität



Die Rolle des Tests im Entwicklungsprozess

Integrationstest

- Auch Interaktionstest
- Mehrere Komponenten werden schrittweise integriert und zusammen getestet
- Es gibt verschiedene Integrationsstrategien, wie z.B.:
 - Top-Down-Integration
 - Ad-hoc-Integration
 - Backbone-Integration
- Testtreiber, Mocks bzw. Simulatoren notwendig
- Ziel: Aufdecken von Schnittstellenfehlern

Die Rolle des Tests im Entwicklungsprozess

Systemtest

- Test des Gesamtsystems
- Fachlicher Test (vorherige Teststufen eher gegen technische Spezifikationen)
- Black-Box-Test
- Häufiges Problem: schlechte Dokumentation der fachlichen Anforderungen
- Ziel: Validierung, ob und wie gut die (funktionalen und nicht funktionalen) Anforderungen erfüllt sind



Die Rolle des Tests im Entwicklungsprozess

Abnahmetest

- Wird i.d.R. vom Auftraggeber durchgeführt
- Typische Formen:
 - Test auf vertragliche Akzeptanz
 - Test auf Benutzerakzeptanz
 - Akzeptanz durch Systembetreiber
 - Feldtest (Alpha- und Beta-Tests)





Grundlegende Testarten

Funktionaler Test

Prüfung auf funktionale Anforderungsmerkmale wie Korrektheit und Vollständigkeit.

Nicht funktionaler Test

Überprüfung der nicht funktionalen Anforderungen, wie Sicherheit, Benutzbarkeit, Dokumentation ...

Strukturbezogener Test

basiert auf der internen Struktur bzw. Architektur der Software.
Analyse z.B. von Kontrollfluss oder Aufrufhierarchien.

Änderungsbezogener Test

oder Regressionstest. Wiederholung von Tests nach Wartung und Pflege.



Statischer Test

Reviews

- **Technisches Review**
Fachliche Prüfung eines wesentlichen Dokumentes (z.B. Architekturentwurf) auf Übereinstimmung mit Spezifikation
- **Informelles Review**
Entspricht inhaltlich dem technischen Review, es soll ihm gegenüber aber Zeit gespart werden und daher wird es als nicht formaler Prozess durchgeführt.
Pair-Programming, Buddy-Testing und Code-Swaps können als informelles Review angesehen werden.
- **Walkthrough**
Diskussion von Szenarien, Probeläufen und Alternativen im Kreis gleichgestellter Mitarbeiter mit möglichst niedrig gehaltenem Aufwand
- **Inspektion**
Formalste Reviewtechnik mit einem dokumentierten Vorgehen nach IEEE 610, IEEE 1028.



Statischer Test

Statische Analyse

- Compiler ist ein statisches Analysewerkzeug
- Prüfung der Einhaltung von Konventionen und Standards
- **Datenflussanalyse**
Zur Aufdeckung von Datenflussanomalien, wie referenzierende Verwendung von uninitialisierten Variablen.
- **Kontrollflussanalyse**
Erstellung von Kontrollflussgraphen (Werkzeugunterstützung nötig) zur Aufdeckung von Anomalien die nicht zwangsläufig Defekte sein müssen aber den Grundsätzen der strukturierten Programmierung widersprechen.
- **Metriken** zur Messung von Qualitätsmerkmalen



Dynamischer Test

Blackbox-Verfahren (ohne Kenntnis des Codes)

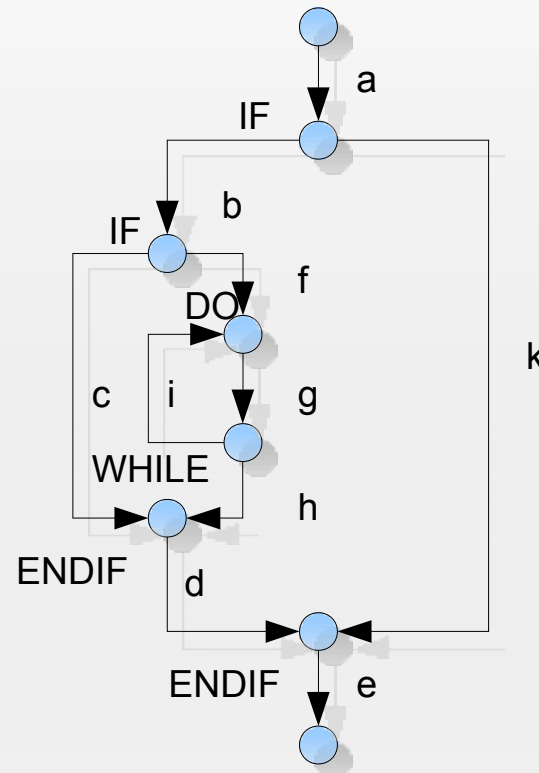
- Äquivalenzklassenbildung
Die Bildung von Testfällen zu (gültigen und ungültigen) Äquivalenzklassen folgt dieser Abfolge:
 - Analyse und Spezifikation der Eingabedaten, der Ausgabedaten und der Bedingungen gemäß den Spezifikationen
 - Bildung der Äquivalenzklassen durch Klassifizierung der Wertebereiche für Ein- und Ausgabedaten
 - Bestimmung der Testfälle durch Wertauswahl für jede Äquivalenzklasse
- Grenzwertanalyse
Testfälle werden aus den Grenzwerten der Äquivalenzklassen gebildet
- Zustandsbezogener Test
Testfälle werden anhand gültiger Zustände und Zustandsübergänge ermittelt. Hilfsmittel: Zustandsdiagramme und Übergangsbäume.
- Ursache-Wirkungs-Graph-Analyse und Entscheidungstabellentechnik
- Anwendungsfallbasierter Test



Dynamischer Test

Whitebox-Verfahren

- Anweisungsüberdeckung (Knoten stehen im Mittelpunkt)
- Zweigüberdeckung (Kanten stehen im Mittelpunkt)
- Bedingungsüberdeckung (Prüfung zusammengesetzter Bedingungen)
- Pfadüberdeckung (oft nicht erreichbar) (alle möglichen Pfade müssen durchlaufen werden, also auch alle möglichen Schleifenwiederholungen)





Testwerkzeuge

Für Management und Steuerung

- Werkzeuge zur Verfolgung der Testfälle und Testdurchführung
- Bugtracking
- Konfigurationsmanagement (Versionierung)
- Beispiele:
 - Testbench
 - Bugzilla
 - Subversion



Testwerkzeuge

Zur Testspezifikation

- Testgeneratoren
 - Datenbankbasiert
 - Codebasiert
 - Schnittstellenbasiert
 - Spezifikationsbasiert



Testwerkzeuge

Für statischen Test

- Zur Reviewunterstützung, z.B. Verwaltung von Checklisten und Dokumentation der Ergebnisse
- Statische Analysatoren zur Lieferung von Code-Metriken und Überprüfung auf Einhaltung von Style-Guides, Syntax-Prüfung (Compiler) usw.
- Model Checker zur Validierung der Implementierung gegen ein formales Modell



Testwerkzeuge

Für dynamischen Test

- Debugger
- Testtreiber
- Simulatoren
- Testroboter (Capture-and-Replay)
- Testframeworks (xUnit)



Was fehlt?

Testmanagement

- Testorganisation
- Testplanung
- Kosten- und Wirtschaftlichkeitsaspekte
- Wahl der Teststrategie
- Management der Testarbeiten
- Fehlermanagement
- Anforderungen an das Konfigurationsmanagement
- Relevante Normen und Standards



Quellen

- [Spillner 07] Spillner, A.; Linz, T.: Basiswissen Softwaretest.
dpunkt.verlag, Heidelberg, 2005 (3. Aufl., korr. Nachdruck 2007)
- [Balzert 98] Balzert, H.: Lehrbuch der Software-Technik: Software-Management,
Software-Qualitätssicherung, Unternehmensmodellierung
Spektrum, Akad. Verl., Heidelberg, 1998
- [Rätzmann 04] Rätzmann, M.: Software-Testing & Internationalisierung
Galileo Press GmbH, Bonn 2004
- [URL: WIKI] <http://de.wikipedia.org/>
- [URL: ISTQB] <http://www.istqb.org/>
- [URL: GTB] <http://www.german-testing-board.info>
- [URL: JUNIT] <http://www.junit.org/>



Noch Fragen?

