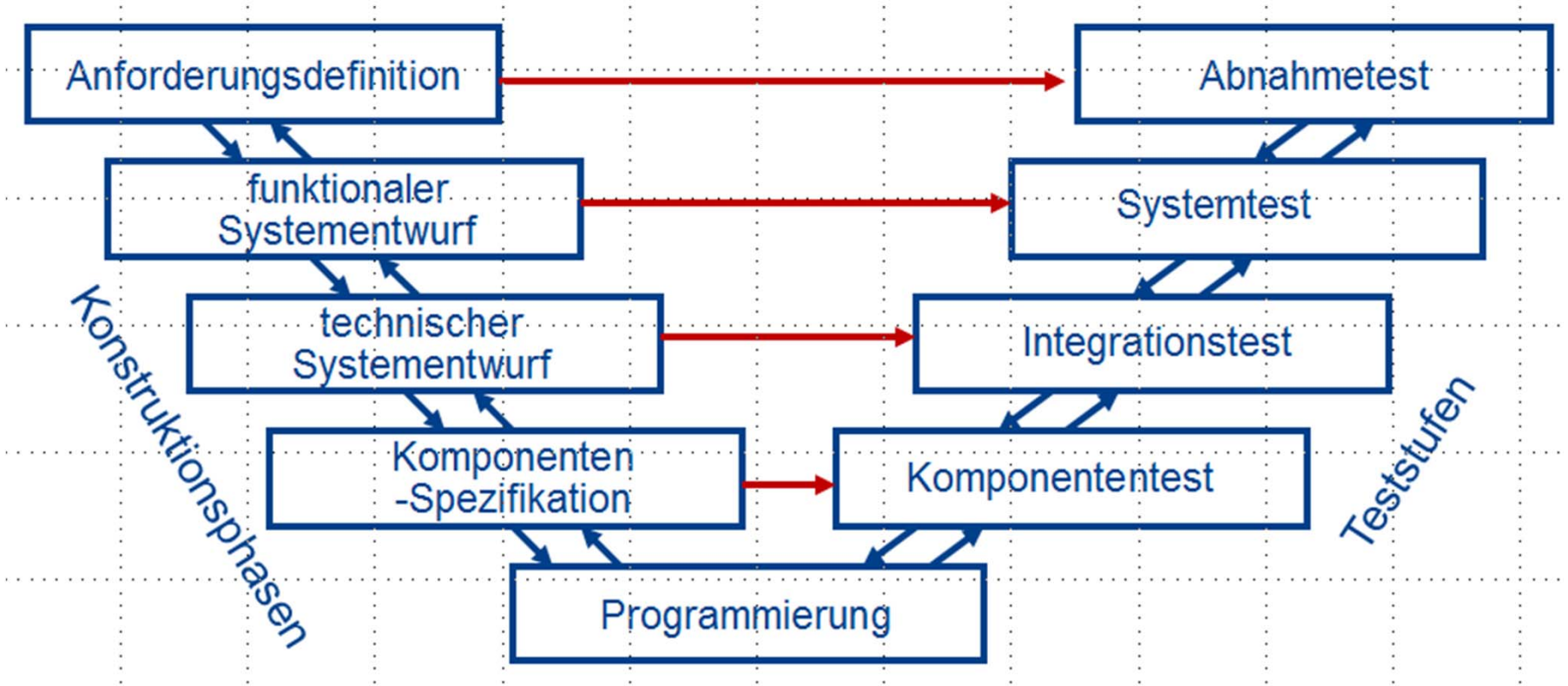
SOFTWARE-ENGINEERING 2

TESTEN IM ENTWICKLUNGSZYKLUS

Übersicht


Anforderungsdefinition
Abnahmetest
funktionaler Systementwurf
Systemtest
technischer Systementwurf
Integrationstest
Komponenten -Spezifikation
Komponententest
Programmierung
Konstruktionsphasen
Teststufen

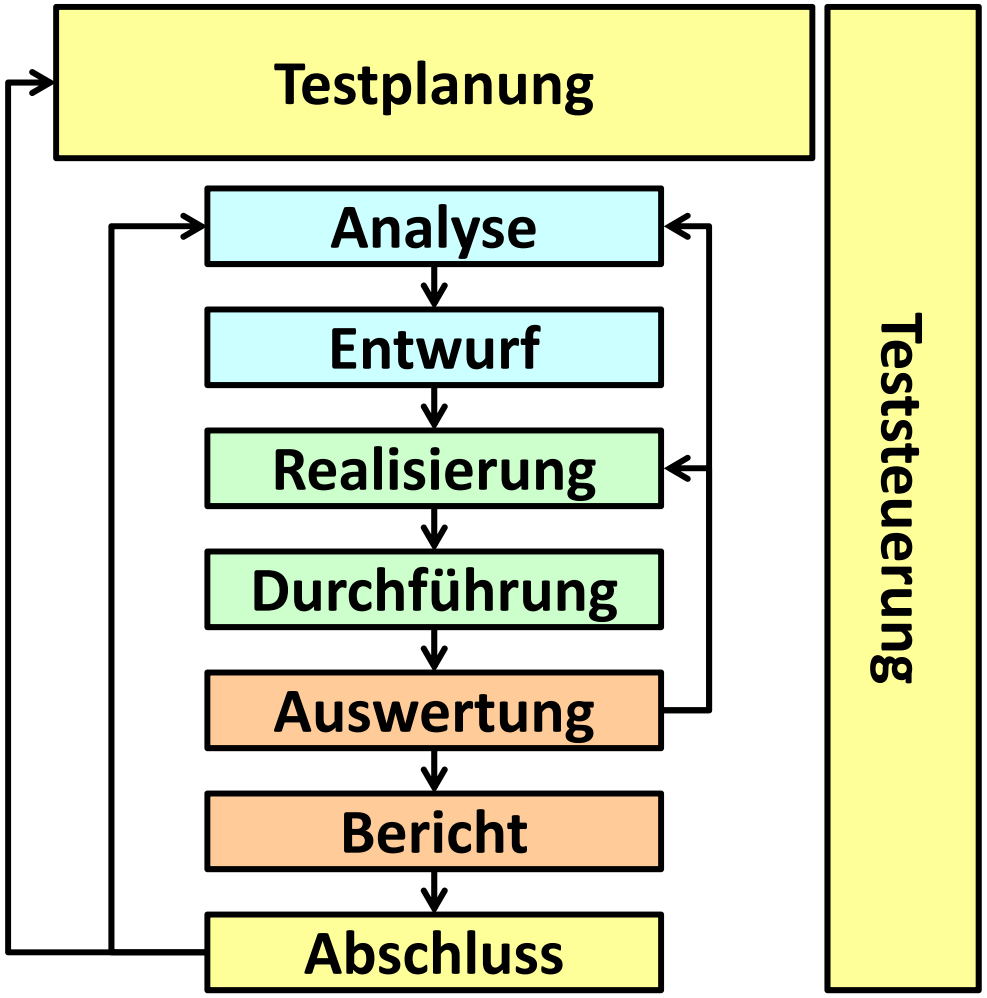
Einordnung des Testens

- Die Testaktivitäten sind im V-Modell auf dem „rechten Ast“ den Aktivitäten des „linken Astes“ nachgeordnet
  - Dies gilt nur inhaltlich!
  - Ohne eine Beschreibung des Sollverhaltens kann kein Test erfolgen.
  - Zeitlich können Testplanung, -analyse und –entwurf parallel zu den anderen Entwicklungsaktivitäten stattfinden.
  - Ohne ein Produkt kann nicht getestet werden
    → Testdurchführung nachgelagert.

- Die verschiedenen Teststufen
  - verfolgen unterschiedliche Ziele
  - verwenden unterschiedliche Werkzeuge
  - werden von unterschiedlichen Menschen umgesetzt

- Der Testprozess ist in den Entwicklungsprozess integriert, hat aber eine eigene Organisation

Fundamentaler Testprozess



Teststufen

- **Komponententest**
  Prüfung der SW-Komponente gegen ihre Spezifikation

- **Integrationstest**
  Prüfung des Zusammenwirkens mehrerer SW-Komponenten gemäß Systementwurf (SW-Architektur)

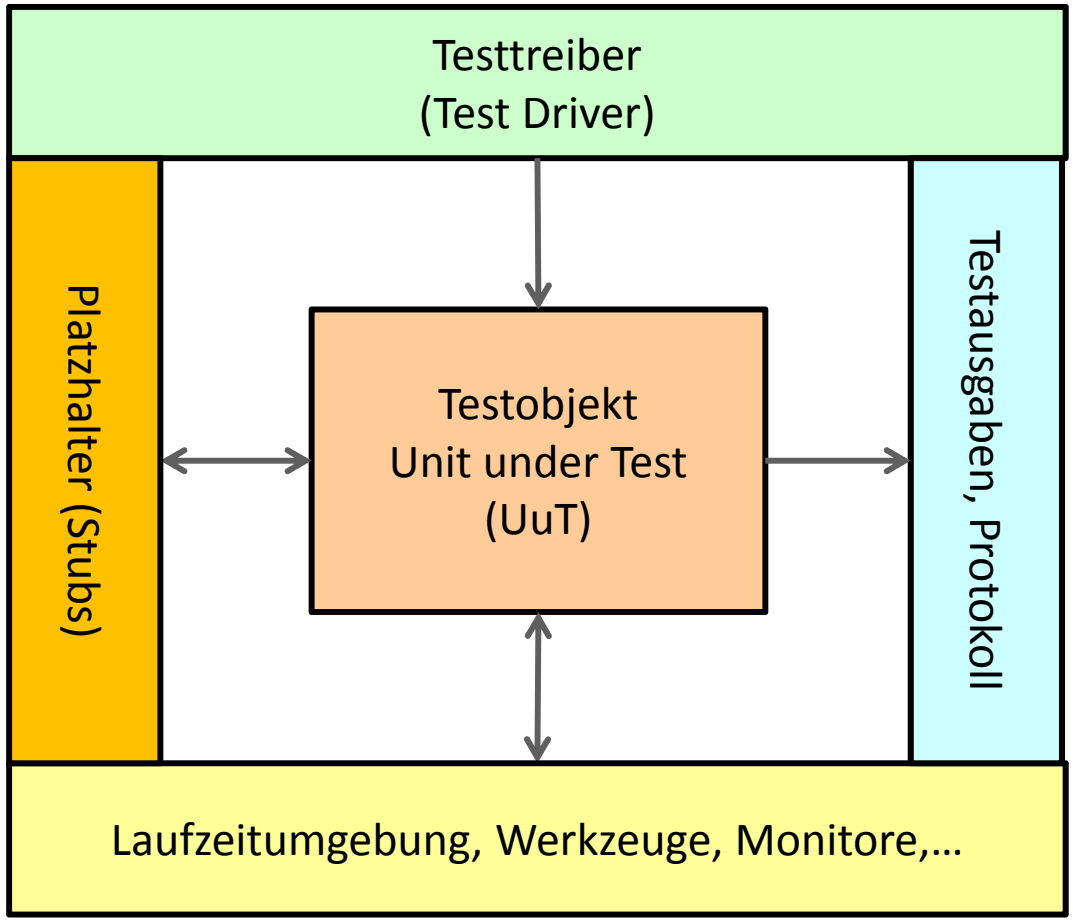
- **Systemtest**
  Prüfung des Gesamtsystems gegen die Anforderungen

- **Abnahmetest**
  Prüfung der vertraglich vereinbarten Leistungsmerkmale

Komponententest – Testobjekt

- Abhängig von der verwendeten Programmiersprache
- Unterschiedliche Bezeichnungen: Klassen, Module, Unit
  Entsprechend: Klassentest, Modultest, Unit Test
- Erster Test nach der Programmierphase
- **Testobjekt**
  Isolierte Komponente
  Keine Wechselwirkungen mit anderen (programmierten) Komponenten
- Fehler lassen sich eindeutig der Komponente zuordnen
- **Testbasis (Orakel)**
  Spezifikation der Komponente
  Programmcode (Vorsicht!)
  Weitere Dokumente, aus denen auf das Verhalten der Komponente geschlossen werden kann (z.B. Lastenheft)

Komponententest – Testumgebung



Komponententest – Testziele

- Der Komponententest soll die Funktionalität der Komponente sicherstellen.
- Funktionalität ist das Ein-/Ausgabeverhalten der Komponente
- Typische Fehlerwirkungen, die beim funktionalen Komponententest aufgedeckt werden, sind Berechnungsfehler oder fehlende und falsch gewählte Programmpfade (z. B. vergessene Sonderfälle).
- Aber auch:
  Robustheit, Wartbarkeit, Effizienz

Test-first-Ansatz

- Idee
  Zunächst alle Testfälle spezifizieren und umsetzen
  Alle Testfälle sind automatisiert
  Anschließend Funktionalität implementieren und fortlaufend testen

- Iterative Entwicklung
  Solange verbessern, bis keine Fehler mehr auftauchen

- Test-Driven-Development
  Zyklen aus Testfallentwicklung, Programmierung, Integration und Test

Integrationstest

- Testen von Komponenten im Zusammenspiel

- **Testobjekt**
  Vorher getestete (und korrigierte) Komponenten werden zusammengebaut (Integration)
  → Aufgabe des SW-Integrators (Rolle)

  Integration kann mehrstufig erfolgen

- **Testziele**
  Sicherstellen des korrekten **Zusammenspiels** der Komponenten
  Entdeckung von Fehlerzuständen in den Schnittstellen der Komponenten

Integrationstest – Testumgebung

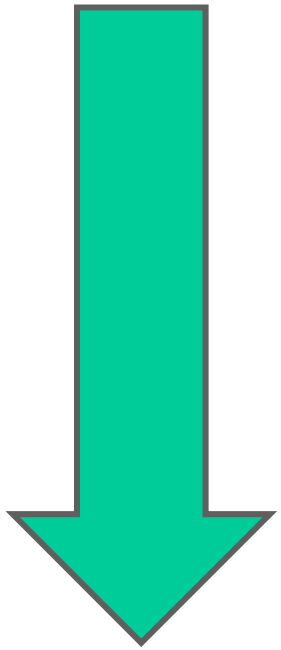
- Oft wie beim Komponententest
- Zusätzliche Monitore notwendig, um die Schnittstellen zu beobachten
- Komplizierte Wechselwirkungen zwischen Komponenten
  → Fehlerzustände sind nur durch dynamische Tests zu finden

- Typen von Fehlerzuständen
  - Protokollfehler
    Schnittstellen werden syntaktisch falsch bedient
  - Semantische Schnittstellenfehler
    Protokoll wird (syntaktisch) richtig bedient, aber die Daten werden unterschiedlich interpretiert
    Beispiel: Auflösung von Wertebereichen
  - Zeitfehler (Timing-Problem)
    Daten werden zum falschen Zeitpunkt (zu früh, zu spät) übermittelt oder die Datenlast ist zu hoch (Durchsatzproblem)

→ Diese Fehlerzustände können nicht im Komponententest gefunden werden, weil die Fehlerwirkung erst mit der Integration zu Tage tritt.

Integrationsstrategie – Top-down

- Zunächst wird die Komponente getestet, die die anderen Komponenten aufruft.
- Andere Komponenten werden durch Platzhalter (Stubs) ersetzt.
- Nach und nach werden (gemäß Aufrufhierarchie → Architektur!) die Komponenten hinzugefügt.

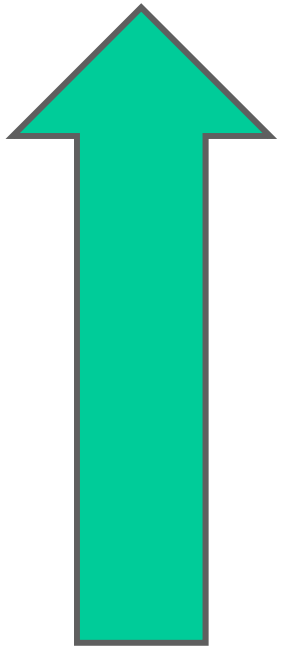


- Testtreiber: Bereits getesteter Integrationsstand

- Problem: Sehr viele Platzhalter

Integrationsstrategie – Bottom-up

- System wird aus einfachen Komponenten nach und nach zusammengebaut
- Anfang: Komponenten, die keine anderen mehr aufrufen
- Vorteil: Keine Platzhalter notwendig
- Nachteil: Übergeordnete Komponenten müssen durch Testtreiber ersetzt werden



- In der Praxis werden Mischformen angewandt, da die meisten Architekturen nicht durchgängig hierarchisch aufgebaut sind.
- Je größer der Integrationsschritt, desto schwieriger ist das Isolieren einzelner Fehlerzustände!

- **Testobjekt**
  Auszuliefernde Produkt → Softwarestand
  Im Fall eingebetteter Systeme → Hard- und Software

- **Testziele**
  Sicherstellen, dass das Produkt die Anforderungen erfüllt.
  - Überprüfen von funktionalen Anforderungen
    → Vorherige Teststufen testen gegen die technische Spezifikation
    → Zusammenspiel mit Umgebung
  - Überprüfen von nicht-funktionalen Anforderungen
    → Bedienbarkeit, Effizienz, Zuverlässigkeit

Systemtest – Testumgebung

- Testumgebung sollte möglichst der Produktivumgebung entsprechen
- Treiber, Stubs → Reale Umgebung (Hardware, Softwareumgebung)

→ aber: Nicht in Produktivumgebung testen
→Mögliche Schäden
→Reproduzierbarkeit
→Kontrollierbarkeit

- Eingebettete Systeme
  Test in realer Umgebung (z.B. Testfahrzeuge, Brettaufbauten)
  Simulation der Umgebung (Hardware in the Loop – HIL)

Systemtest – Teststrategie (1/3)

- Anforderungsbasierte Testfälle
  - Anforderungsdokument dient als Testbasis
  - Systemtestspezifikation wird begutachtet (Review gegen Anforderung)
  - Zu jeder Anforderung (im Lastenheft) wird mindestens ein Testfall im Systemtest spezifiziert (i.a. Positivtest)
  - Um eine Anforderung zu testen werden i.a. mehrere Testfälle benötigt (insbesondere auch Negativtestfälle)

  → Sinnvoll, wenn Anforderungen gut (was heißt das?) spezifiziert sind

Systemtest – Teststrategie (2/3)

- Geschäftsprozessbasierte Testfälle
  - Ausarbeitung der relevanten (und kritischen) Geschäftsvorgänge
    → Geschäftsprozessanalyse (Anforderungsanalyse)
    Beispiel: Buchungen im Warenwirtschaftssystem, Abfrage Kontostand
  - Aufstellen typischer Vorgänge (Szenarien) mit konkreten Daten
    → Testfallspezifikation
    → Sequenzdiagramme
  - Priorität richtet sich nach der Kritikalität der Geschäftsvorgänge

  → Sinnvoll, wenn das SW-System Geschäftsprozesse abbildet

Systemtest – Teststrategie (3/3)

- Anwendungsfallorientierte Testfälle
  - Systemtestfälle ableiten, wie der Anwender mit dem System umgeht und welche Aktionen er dann typischerweise ausführt.
  - Verschiedene Anwendergruppen besitzen jeweils ihre eigenen Benutzerprofile. Das heißt, für sie lassen sich typische Aktionsmuster oder Anwendungsfälle (*use cases*) in typischer Häufigkeit identifizieren.
  - Aus diesen Aktionsmustern lassen sich wieder Testszenarien ableiten.
  - Anhand der Häufigkeit, mit der die entsprechenden Aktionen im späteren Betrieb der Software ausgelöst werden, ermittelt der Tester, wie wichtig das zugehörige Testszenario ist und mit welcher Priorität es deshalb im Testplan aufgenommen werden sollte.

- Lasttests
  - Test des Systems unter Last (z.B. viele Anwender parallel)
  - Prüfung auf Degradation des Verhaltens
- Performanztests
  - Bestimmung der Reaktionszeit des Systems für bestimmte Anwendungsfälle
  - Kopplung mit Lasttests
- Volumen-/Massentest
  - Beobachtung des Systemverhaltens in Abhängigkeit zur Datenmenge (z. B. Verarbeitung sehr großer Dateien)
- Stresstest
  - Beobachtung des Systemverhaltens bei Überlastung

- Test der (Daten-)Sicherheit
  - Versuch des unberechtigten Systemzugangs oder Datenzugriffs

- Test der Zuverlässigkeit
  - Dauerbetrieb unter realen Bedingungen
    (z. B. Ausfälle pro Betriebsstunde bei gegebenem Benutzungsprofil)

- Test auf Robustheit
  - Reaktion auf Fehlbedienung, Fehlprogrammierung, Hardwareausfall usw.
  - Prüfung der Fehlerbehandlung und des Wiederanlaufverhaltens (*recovery*).

- Test auf Kompatibilität/Datenkonversion
  - Prüfung der Verträglichkeit mit vorhandenen Systemen. Import/Export von Datenbeständen usw.

Systemtest – Teststrategie für nichtfunktionale Anforderungen

- Test unterschiedlicher Konfigurationen
  - Unterschiedliche Betriebssystemversion, Landessprache, Hardwareplattform…
- Test auf Benutzungsfreundlichkeit/Benutzbarkeit
  - Prüfung der Angemessenheit der Bedienung
  - Verständlichkeit der Systemausgaben usw.

  → jeweils bezogen auf die Bedürfnisse einer bestimmten Anwendergruppe.
- Prüfung der Dokumentation
  - Übereinstimmung mit dem Systemverhalten
- Prüfung auf Änderbarkeit/Wartbarkeit
  - Verständlichkeit und Aktualität der Entwicklungsdokumente
  - Modulare Systemstruktur
  - Codier-Richtlinien

Abnahmetest

- **Testobjekt**
  Gesamtsystem

- **Testziel**
  Prüfung des Systems gegen vertraglich (und regulatorisch) festgelegte Abnahmekriterien

- **Testumgebung**
  Produktivumgebung des Systems

Abnahmetest – Teststrategie

- Akzeptanztest
  Repräsentative Benutzer
  → Frühzeitig einbinden!

- Alphatest
  Test durch Endanwender beim Softwarehersteller

- Betatest
  Test durch Endanwender
  → Einfluss verschiedener Systemkonfigurationen