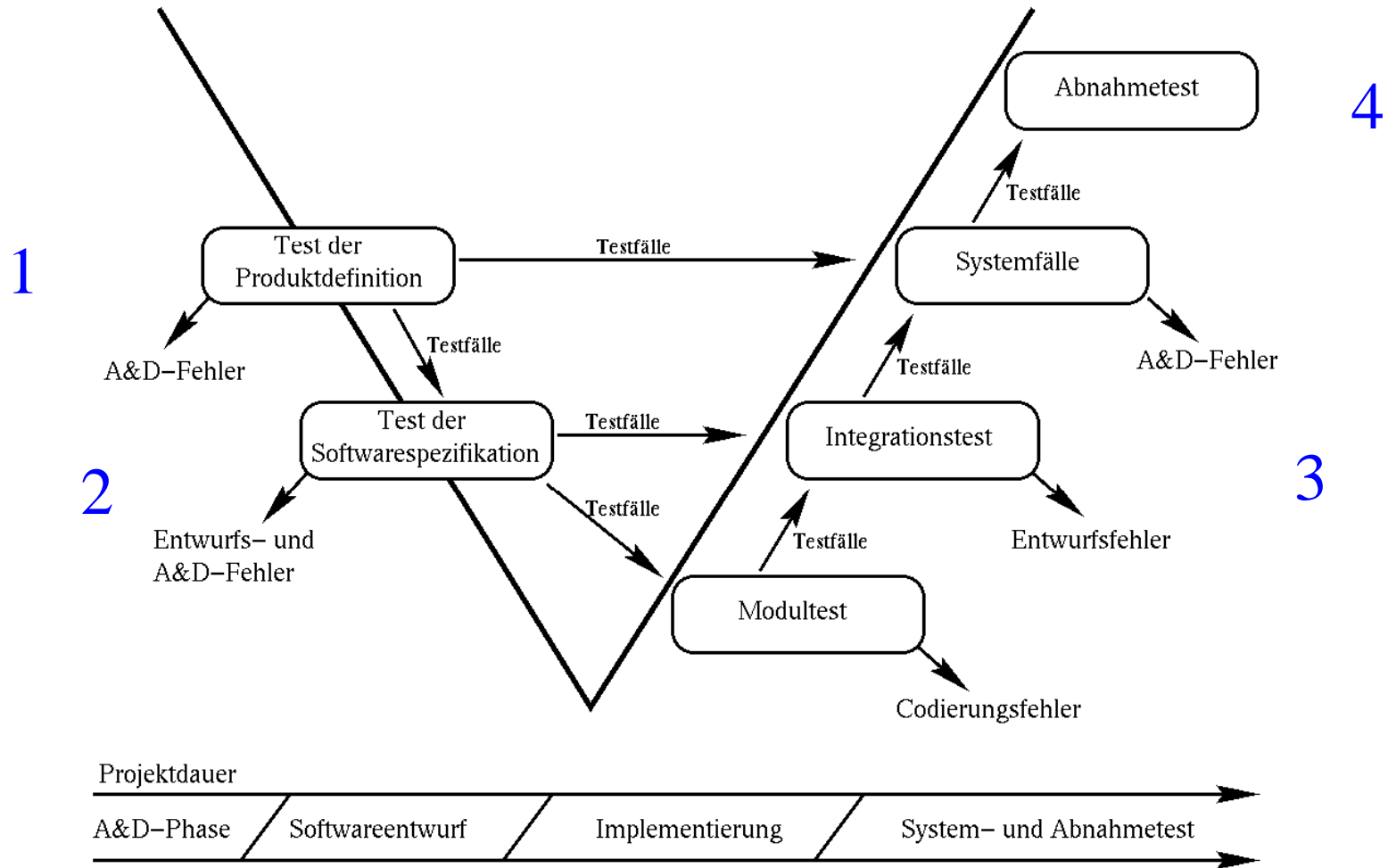


## VL-Inhalte (SS 02)

- Software-Engineering
- Wartung/ Reengineering
  - Definition
  - Beispiele
  - Tätigkeiten
- Qualitätssicherung

# Qualitätssicherung

- *Überblick*
  - *Definition*
  - *Konstruktive und analytische QS*
    - *Planung und Testmethodik*
  - *Analytische QS im Software-Lebenszyklus*
-



# Testverfahren/ Prüfverfahren

Zielsetzung

**verifizierend**

Programmverifikation  
symbolische Programmausführung

**analysierend**

Metriken  
Graphiken und Tabellen

**testend**

**statisch**

- Reviews

**dynamisch**

- strukturorientiert  
kontrollflussbezogen  
- Anweisungsüberdeckung  
- Zweigüberdeckung  
- Bedingungsüberdeckung  
datenflussbezogen  
- Defs/Uses-Kriterien

- funktional  
funktionale Äquivalenzklassenbildung

- diversifizierend  
Mutationen-Test, Pertubations-Test  
Back To Back-Test

- Zufallstest  
- Test spezieller Werte  
- Grenzwertanalyse

Statische Verfahren

Dynamische  
Verfahren

Basis ist das V-Modell von Boehm

1. Test der Produktdefinition
  2. Test der Softwarespezifikation
  3. Test der Implementierung bestehend aus
    - Modultest
    - Integrationstest
    - Systemtest
  4. Abnahmetest
-

## ***Ziel/ zu prüfen:***

ist die Produktdefinition vollständig, eindeutig, widerspruchsfrei; gibt sie die Vorstellungen des Auftraggebers wieder; können Anforderungen kombiniert oder allgemeiner gefasst werden?

Gleichzeitig Sammlung von Testfällen für den Abnahmetest.

## ***Typische Fehler:***

vergessene, falsch oder mehrdeutig beschriebene Funktionen.

## ***Problem:***

In der Praxis häufig am schlechtesten ausgeführt, obwohl hier eingebaute Fehler oft sehr aufwendig zu beheben sind. Abstraktionsniveau muss stimmen.

## ***Verfahren:***

Review, an dem neben Moderator, Testspezialisten, Analytikern auch der Auftraggeber und ein Benutzer teilnehmen. Das Dokument "Produktdefinition" sollte graphisch aufbereitet vorliegen.

---

Weitere Maßnahmen zur QS in dieser Phase sind:

- eine Testorganisation wird aufgebaut
- Testplanung
- Richtlinien vorgeben z.B. bzgl. Dokumentation
- technische Vorarbeiten für die Testaktivitäten u.a.

Installation und Initialisierung von Testhardware,  
Testdatengeneratoren, Testdatenbanken

## *Ziel/ zu prüfen:*

Validieren der Systemarchitektur, Überprüfung der Teilsystem-, Modul- und Prozessspezifikationen.

Sind alle Komponenten und Operationen vollständig, eindeutig und widerspruchsfrei spezifiziert?

Werden die Anforderungen aus der Produktdefinition erfüllt?

Kann die Architektur verbessert werden?

Werden Qualitätskriterien des modularen Systementwurfs verletzt? (Modulanzahl, -größe, hohe Kohäsion, geringe Kopplung)

## *Typische Fehler:*

vergessene, falsch spezifizierte oder anders als vorgesehen spezifizierte Funktionen, Fehler in der Datenspezifikation.

## *Verfahren:*

Zwei Reviews:

nach der Architekturfestlegung und nach dem Feinentwurf

---

Weitere Aktivitäten in dieser Phase sind:

- Systemtest planen und Testfälle (Black-Box-Test) sammeln
- Integrationstest planen
- Modultest planen und Testfälle sammeln für Modultest (Black-Box)
- Für den Integrations- und den Modultest müssen Treiber und Stubs vorgesehen werden, d.h. geeignete Testumgebungen müssen entwickelt werden

*Treiber:*

steuert Prüfling über exportierte Prozeduren an und versorgt ihn mit Testdaten (z.B. über Tastatur)

*Stubs:*

implementieren Exporte von benutzten Modulen, die bisher noch nicht getestet wurden

---



## *Ziel/ zu prüfen:*

- Test der exportierten und lokalen Prozeduren eines Moduls.
  - Funktionalität, Struktur und Performanz der Prozedur testen.
  - Untersuchung der Qualität der Zerlegung im Modulrumpf (z.B. mehrfach codierte Teile in lokale Hilfsprozeduren zusammenfassen)
  - werden alle definierten Konstanten, Variablen, Datentypen auch benutzt
  - werden alle Importe benutzt
  - Festlegung einer Testreihenfolge (bottom-up)
  - Test des Moduls gegen die Spezifikation, d.h. stimmen Spezifikation und Implementation überein.
-

## *Typische Fehler:*

- Kontrollflussfehler,  
z.B. falsch durchlaufene Pfade, fehlende Pfade (Sonderfälle vergessen)
- Datenflussfehler
- Funktionale Fehler
- Konstanten-/ Operatorenfehler, z.B. Berechnungs-, Datenfehler
- Fehler in der Lösungsmethode

## *Verfahren*

- Verwendung statischer Programm-Analysatoren
  - Code-Review (systematisches Lesen des Quellcodes):
    - billig und effektiv (bis zu 70% der gefundenen Fehler bereits hier)
  - dynamische Tests
-

## Minimalanforderung:

- mit Black-Box-Verfahren  
Code/ Spezifikation testen
- mit White-Box-Verfahren  
vollständige Zweigüberdeckung anstreben

## Allgemein:

funktionalen (Black-Box) dem strukturorientierten (White-Box) Test vorziehen

---

Sukzessives Zusammenmontieren getesteter Komponenten.  
Die Reihenfolge wird durch die Integrationsstrategie festgelegt  
(sie hat großen Einfluss auf die nötigen Treiber bzw. Stubs).

nicht - inkrementell		Big Bang transaktionsorientiert
inkrementell	bottom up top down inside out hardest first outside in	funktionsorientiert nach Verfügbarkeit
	vorgehensorientiert	zielorientiert

Integrationen sind

- vorgehensorientiert:

aus dem Komponentengraph abgeleitet

- zielorientiert:

Komponenten werden bzgl. konkreter Testfälle/ -ziele  
zusammengesetzt

- inkrementiell:

einzelnen oder in kleinen Gruppen werden Module zusammengefügt

- nicht-inkrementiell:

alle Komponenten oder große Gruppen werden gleichzeitig  
zusammengesetzt. Nicht-inkrementiell ist für große Software  
ungeeignet!

---

- Big Bang: alle Module gleichzeitig
  - transaktionsorientiert: alle zu einer bestimmten Transaktion nötigen Module werden integriert (z.B. Kunde einrichten)
  - nach Verfügbarkeit: Implementationsreihenfolge bestimmt Integration
  - funktionsorientiert: Integration nach funktionalen Testfällen (aber schrittweise)
  - top-down: beginnend mit der Benutzungsschnittstelle aufwendig, da viele stubs nötig
  - bottom-up: beginnend mit "Basis"-Moduln; lauffähige Version erst am Ende
  - inside-out: beginnend auf einer mittleren Ebene werden Module nach oben und unten hin dazugefügt.
  - outside-in: es wird gleichzeitig auf der höchsten und niedrigsten Schicht begonnen: Kompromiss zu top-down und bottom-up.
  - hardest-first: die kritischen Komponenten zuerst (z.B. die, die potentiell am meisten Fehler haben)
-

## *Ziel/ zu prüfen:*

- Überprüfung des Außenverhaltens des Systems
- Test der operativen Anforderungen und der Schnittstellen zu externen Systemen
- Leistungstests:
  - Laufzeittests validieren Effizienzanforderungen
  - Stresstests
    - testen Spitzenbelastungen
    - testen mit großen Datenmengen (Massentest)
- Validieren der Benutzungsfreundlichkeit
- Validieren der Wartbarkeit
- Validieren der Dokumentation
- Validieren der Zugangsschutzmechanismen.

## *Verfahren:*

- funktionale Tests
  - statische Tests für die Dokumentation
-

## *Ziel/ zu testen*

- der Abnahmetest stellt die abschließende Validierung des Systems mit realen Daten auf der Zielplattform dar
- muss portiert werden, so erfolgt danach noch ein Installationstest.

## Verfahren:

- funktionale Tests.

- stellt unabhängige Testtätigkeit dar
- Testfälle sind Teilmenge der Testfälle des Systemtests
- Testfälle sind zum großen Teil schon in der Produktdefinition spezifiziert

## *Methodisches Vorgehen*

Alpha-Test: beim Auftraggeber unter dessen Regie,  
falls Auftrags-Software erstellt wurde

Beta-Test: falls Produkt-Software erstellt wird:

ausgewählte Kunden erhalten Pilotsystem zur Probebenutzung,  
Probleme und Fehler werden protokolliert (Bug-Report) und an den Anbieter  
zurückgemeldet

---

Testwerkzeuge liefern

- Metriken
- Graphiken
- Einfache Überdeckungstests
- Aussagen zu Speichernutzung

Quellen/ Systeme

- OVUM Reports

# Testwerkzeuge (OVUM-Reports)

**(Stand: 15.6.98)**

Aonix	StP/T
Autotester	Autotester
Bender & Associates	SoftTest
CenterLine Software	QC/Advantage, QC/Replay, C++ Expert
<b>Compuware</b>	QACenter, QA Hiperstation
<b>Cyrano</b>	CYRANO Suite, V-TEST
Dassault Electronique	DEVISOR
Elverex	Evaluator FT
Hammer Technologies	Hammer
Imago QA	T-Plan
<b>IPL</b>	Cantata
LDRA	LDRA Testbed
Mainsoft	MainWin Test
<b>McCabe &amp; Associates</b>	The McCabe Toolset
<b>Mercury Interactive</b>	Astra SiteManager, Astra SiteTest, LoadRunner, TestBytes, TestDirector, TestSuite 2000, WinRunner, XRunner
<b>MuTek</b>	Bugtrapper
Programming Research	QAC, QAC++
Qronus Interactive	TestRunner
<b>RadView</b>	Webload
<b>Rational Software</b>	SQA Suite, preVue-C/S, preVue-X, PurePerformance/CS, Pure DDTs, Pure Coverage, Purify, Quantify, TestMate, Visual Test
<b>Reliable Software Tech.</b>	DeepCover, SafetyNet, AssertMate, TotalMetric
<b>Segue Software</b>	Quality Works, SilkTest
Silicon Valley Networks	TestExpert
Softbridge	ATF
Software Research	Software TestWorks
Teradyne	Testmaster
Verilog	Logiscope
VisionSoft	VisionSoft

**(Stand: 1.12.2000)**

- ATTOL
- Compuware
- Cyrano
- IPL
- McCabe and Associates
- Mercury Interactive
- MuTek Solutions
- The Original Software Co
- Radview Software
- Rational Software
- RSW Software
- Segue Software
- Technology Builders
- TeleLogic
- T-Plan
- Worksoft

**(Stand: 12.11.2001)**

- Compuware
- McCabe and Associates
- Mercury Interactive
- Radview Software
- Rational Software
- Empirix (previously RSW Software)
- Segue Software
- TeleLogic
- T-plan

## Zur Qualitätssicherung

- Balzert, H.: Lehrbuch der Software-Technik, Spektrum, 1998
  - Sommerville, I.: Software Engineering, 6.Auflage, Addison-Wesley, 2001
  - Liggesmeyer: Modultest und Modulverifikation, BI, 1990
  - Liggesmeyer: Wissensbasierte Qualitätsassistenz, BI, 1993
  - Frühauf, K. et al.: Software-Prüfung, Teubner/ vdf, 2. Aufl., 1995
  - Lindermeier/ Siebert: Softwareprüfung und Qualitätssicherung, Oldenbourg, 1995
  - Wallmüller, E.: Software-Qualitätsmanagement in der Praxis, Hanser, 2001
-