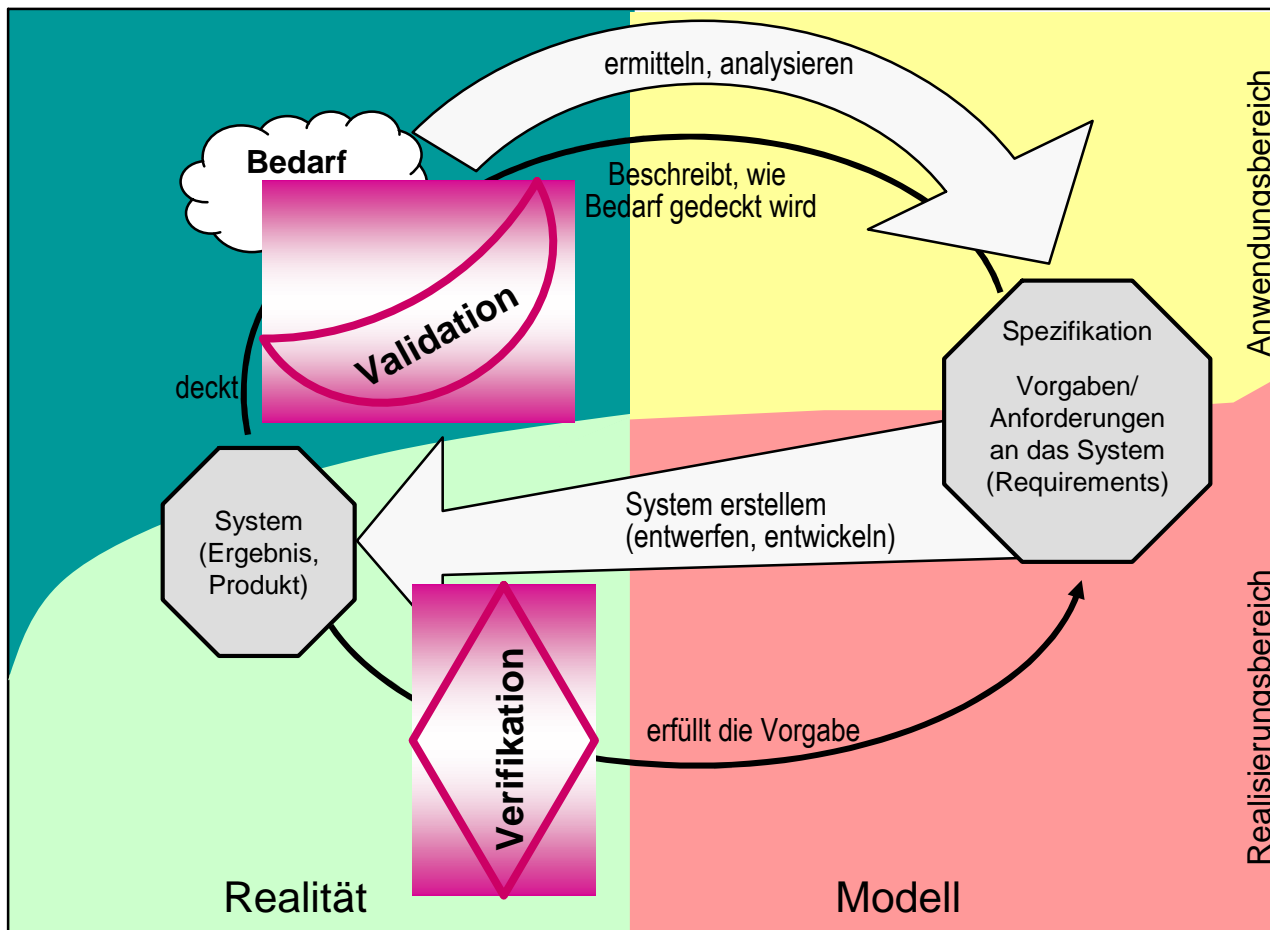


Überblick über die Prüfverfahren

Was ist Validieren und Verifizieren?



Validieren:

Tut das System das Richtige?

Ein Fachmann/frau prüft die Spezifikation oder das gesamte System

Verifizieren:

Tut das System das richtig?

Gegen die Spezifikation prüfen.

Die **Validation** untersucht die Frage
Wird das richtige Haus gebaut?

Die **Verifikation** untersucht die Frage
Wird das Haus richtig gebaut?

Testphasen

- Die Applikation wird nicht als Ganzes getestet, sondern kleinere Einheiten (Module, Funktionen).
- Dies ist vorteilhaft, da man den Ort des Fehlers besser bestimmen kann. Außerdem muß man beim Einzeltest nicht alle Eingangsbedingungen kombinieren.
- Durch die Möglichkeit des parallelen Tests ergibt sich auch ein Zeitvorteil.

Ziele:

- Aufdecken von Fehlern (Programmierfehler, funktionale Fehler) und Nachweis der technischen Korrektheit der Implementierung
- Nachweis, daß das Modul die spezifizierten Anforderungen erfüllt
- Erreichen einer vorgegebenen Testabdeckung
- ggf. Nachweis der Robustheit in Grenzsituationen ("Streß-Tests")

- Integration der Module zu Komponenten.
- Wird oft als auch als Integrationstest bezeichnet.
- Auch hier sind parallele Tests möglich.
- Fehlende Module, die noch nicht zur Verfügung stehen, werden durch Testtreiber bzw. Stubs ersetzt.

Ziele:

- Aufdecken von Fehlern in den Schnittstellen zwischen Modulen
- Nachweis der Integrierbarkeit der Komponente und Aufdecken von Fehlern im Zusammenwirken der Module
- Integration der jeweiligen Module zur Komponente
- Nachweis, daß die Komponente die spezifizierten Anforderungen erfüllt

- Ist der abschließende Test in der realen Umgebung ohne den Auftraggeber.
- Es sind nur die externen Schnittstellen sichtbar.
- Basis für den Test sind Pflichtenheft, Produktmodell, Konzept der Benutzeroberfläche und Benutzerhandbuch.
- Nutzt Techniken aus dem Komponententest, unter Einbeziehung zugekaufter oder externer Komponenten.

Ziele:

- Aufdecken von Fehlern in Schnittstellen zwischen internen und externen Komponenten.
- Nachweis der funktionalen Korrektheit entsprechend der Spezifikation für das System.
- Vorbereitung und Sicherstellen der Abnahme des Systems.

Ist der Test auf vertragliche Akzeptanz.

Beschreibt die letzte Testphase.

Das System wird vom Kunden oder mit dem Kunden getestet.

- Alpha-/ Beta-Tests
- Benchmarks

Prüfverfahren

Statische Prüfungen

- Statische Prüfungen sind Verfahren, die eine Bewertung des Codes vornehmen, ohne ihn auszuführen.
- Unterschieden werden manuelle Prüfungen wie Fagan Inspektion und Walkthroughs, sowie automatische Prüfungen wie z.B. LINT, Polyspace.

Dynamische Tests

- Der zu testende Code wird ausgeführt.
- Dynamische Tests werden unter Berücksichtigung der inneren Struktur (White-Box-Test) bis hin zur Betrachtung als geschlossenes System durchgeführt (Black-Box-Test).

Fagan Inspektion

Verifikation der Software.

In der Vorbereitung wird der Code am Schreibtisch studiert.

In einer gemeinsamen Sitzung wird die Software besprochen und Fehler werden identifiziert.

Teaminspektion

Die Prüfung und Befunderhebung erfolgt individuell durch Gutachter in der Vorbereitungsphase

Die Review-Sitzung dient nur dem Zusammentragen und Bewerten der Befunde. Dubletten und falsche Befunde werden eliminiert.

N-Individuen-Inspektion

Die Prüfung und Befunderhebung erfolgt vollständig individuell durch die Gutachter.

Es gibt keine gemeinsame Sitzung.

Selbstinspektion

Erfolgt wie die N-Individuen-Inspektion, aber Autor ist sein eigener und einziger Gutachter.

Sie ist Effektiver als einfaches Durchlesen, da sie auf rigorosen Prüfverfahren basiert (Checkliste).

Walkthrough

Die Vorbereitung und Durchführung ist ähnlich der Fagan-Inspektion.

Der Unterschied liegt an der Herangehensweise:

Die Teilnehmer entwickeln kleine Testfälle, die sie gedanklich auf den Code anwenden (Paper Test Case).

Dadurch werden aber immer nur Module des Softwarepaketes betrachtet.

Compiler

Überprüfung des Codes auf Übersetzbarkeit.

Voraussetzung für alle anderen statische Verfahren.

Lint

Prüfung des Codes auf Programmierrichtlinien.

Erweiterte Semantikprüfungen.

Statische Beweisverfahren

Mathematisches Prüfverfahren

Identifiziert anhand des Codes ohne dessen Ausführung

- Speicherlöcher
- Nullpointer
- nicht erreichbaren Code
- Deadlocks
- Fehler in Ausdrücken

White-Box-Test

- *Grundlage ist die innere Struktur des Codes.*
 - Anweisungsüberdeckungstest
 - Zweigüberdeckungstest
 - Bedingungsüberdeckungstest
 - Pfadüberdeckungstest

Black-Box-Test

- Erfolgt ohne Kenntnis der inneren Struktur (Schnittstellentest).
 - Fehlererwartung (error guessing)
 - Funktionale Äquivalenzklassenbildung
 - Grenzwertanalyse
 - Zufallstest
 - Zustandsautomaten

Gray-Box-Test

- Kombiniert die Methodik von Black-Box- und White-Box-Tests.

Anweisungsüberdeckungstest

Jeder Befehl im Programm muß mindestens einmal ausgeführt werden.

Zweigüberdeckungstest

Alle Entscheidungen oder Sprünge werden erfaßt (Entwurf entsprechend vieler Testfälle ist nötig). Er enthält den Anweisungsüberdeckungstest vollständig

Bedingungsüberdeckungstest

Test aufgrund der Bedingungen von Schleifen und logischen Ausdrücken. Alle Bedingungen, die zum Durchlaufen eines Zweigs führen können, werden getestet.

Pfadüberdeckungstest

Ziel ist die Ausführung aller unterschiedlichen Pfade des Programms.

Fehlererwartung (error guessing)

- Man legt eine Liste möglicher Fehler oder fehlerträchtiger Situationen an und definiert daraufhin Testfälle.
- Dieses Verfahren stützt sich vor allem auf eine gut gepflegte Wissensdatenbank und auf erfahrene Tester.

Funktionale Äquivalenzklassenbildung

- Für dieses Testverfahren werden die Definitionsbereiche der Eingabeparameter und die Wertebereiche der Ausgabeparameter in Äquivalenzklassen zerlegt.
- Es wird angenommen, daß das Programm für jeden Repräsentanten einer Klasse gleich reagiert.

Grenzwertanalyse

- Testfälle, die die Grenzwerte von Äquivalenzklassen abdecken, sind besonders effektiv, um Fehler zu finden.
- Die Grenzwertanalyse bedient sich der Elemente aus den Äquivalenzklasse, die den Rand der Äquivalenzklasse testen.

Zufallstest

- Tester können dazu neigen, Testfälle zu entwerfen, die mit einer geringen Wahrscheinlichkeit Fehler im Programm entdecken. Dem kann mit einem ergänzenden Zufallstest im bestimmten Maß entgegengewirkt werden.
- Als einzige Testmethode ist der Zufallstest nicht geeignet.

Zustandsautomaten

- Liegt ein Zustandsautomat vor, können Testfälle aus diesem abgeleitet werden. Das Ziel bei diesem Black-Box-Testverfahren ist es, mindestens einmal jeden Zustandsübergang zu durchlaufen.

Vergleich von Code-Prüfverfahren

Kriterien	Black-Box-Test	White-Box-Test	Statische Analyse
Zum			
• verifizieren	(X)	X	X
• validieren	X	./.	./.
von			
• Code	X	X	X
• anderen Teilprodukten, die sich in Graphen abbilden lassen	./.	./.	X
bezüglich			
• erbringen der geforderten Funktionalität	X	(X)	(X)
• unerwünschter Funktionalität	./.	(X)	X
• handwerklich solider Ausführung	./.	./.	X
Anwendbar ohne arbeitsfähiges HW-SW-System	./.	X	X
Anwendbar ohne lauffähigen Code (oder Treiber / Stubs)	./.	./.	X
Kommt ohne zusätzliches Verfahren aus	X	<ul style="list-style-type: none"> • Statische Analyse oder • Informelles Verfahren, zum • Module identifizieren, • Abdeckung beurteilen. 	X
Effizienz [(Zeit-) Aufwand pro gemessenem Kennwert]	schlecht	schlecht	sehr gut

Mit freundlicher Genehmigung von CATS, Dr. Glöe

Leistungstest

Dient zur Überprüfung der Quality of Service.

- Massentest (Volumentest)
Dient zum Test der zu verarbeitenden Datenmengen.
- Zeittest
Überprüft die Einhaltung des spezifizierten Laufzeitverhaltens.
- Lasttest
Er prüft das Verhalten des Systems, wie es auf Ausfälle von Hardware- und Softwarekomponenten reagiert
- Streßtest
Überprüft das Verhalten beim bewußten Überschreiten von definierten den Grenzen.

Funktionstest

Prüfung, ob das Programm alle geforderten Funktionen beinhaltet.

Prüfung der Korrektheit der Realisierung.

Benutzbarkeitstest

Paßt die Benutzerschnittstelle zum Pflichtenheft?

Ist die Benutzerschnittstelle mit dem Wissen des Endbenutzers bedienbar?

Interoperabilitätstest

Test des Zusammenwirkens mit anderen Programmen (TCP/IP, CAN-Bus, ...)

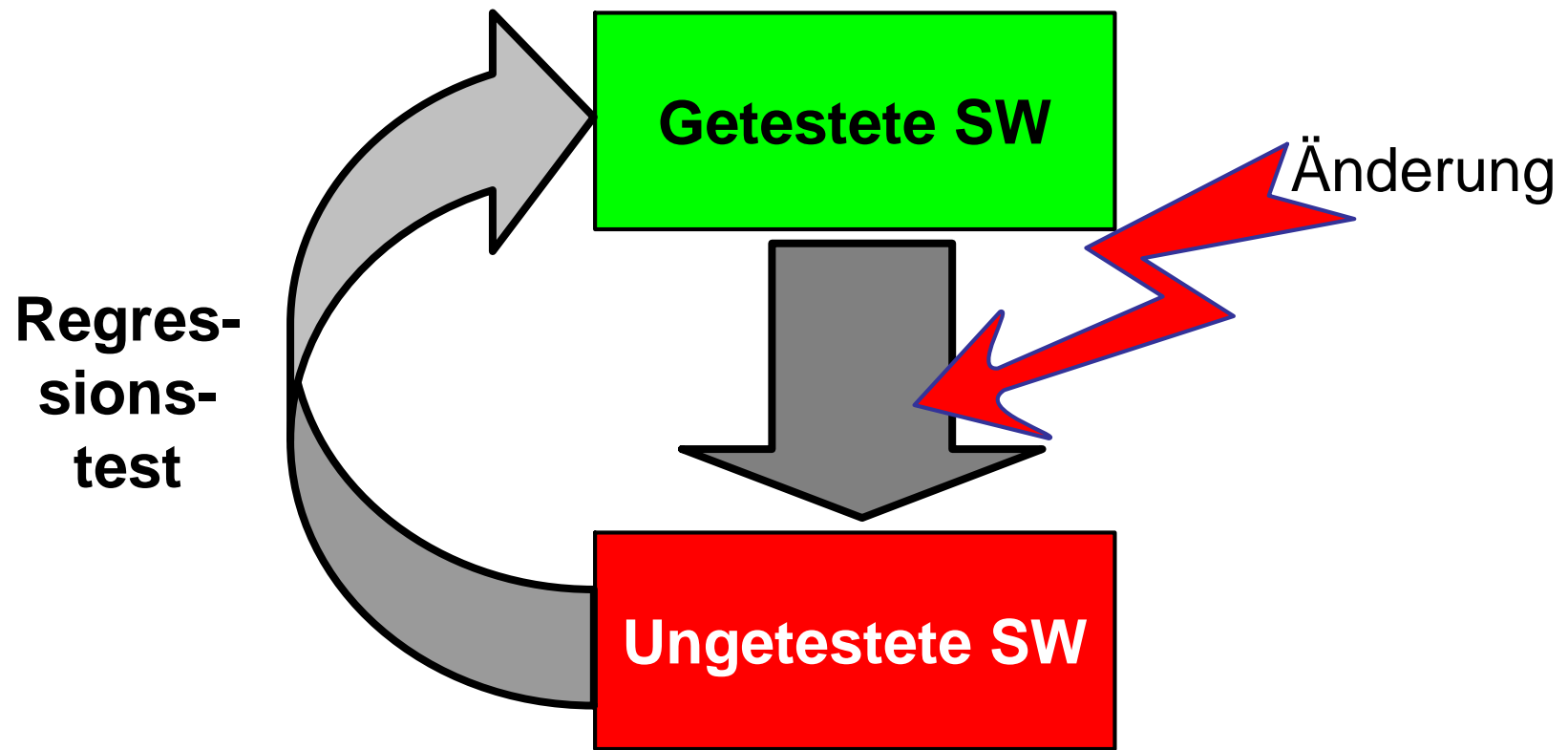
Sicherheitstest

Testen von Zugriffsrechten.

Test von Verschlüsselungen.

Installationstest

Läßt sich das System, wie in der Benutzeranleitung beschrieben, installieren und in Betrieb nehmen?



Ziel: Nachweis, daß alle schon getesteten Funktionen nach einer Codeänderung noch korrekt ablaufen.

Nach einer Codeänderung darf keine Regression auftreten, d.h., die geänderten Anteile müssen weiterhin der Spezifikation genügen.

Dies wird durch **Regressionstests** geprüft.

- Keine eigenständige Testart.
- Wirkt als Klammer über alle Testverfahren.
- Änderungen an existierendem Code können
 - *korrektiv* (Spezifikation verletzt),
 - *inkrementell* (Spezifikation erweitert),
 - *adaptiv* (Spezifikation geändert)
 - oder *optimierend* (Spezifikation unverändert) sein.