



# GTB

German Testing Board

Software. Testing. Excellence.



## Basiswissen Softwaretest Certified Tester Black-Box-Testverfahren

HS@GTB  
2019  
Version 3.1



## Lernziele: Nach dieser Vorlesung sollten Sie ...

- Gemeinsamkeiten und Unterschiede zwischen Black-Box-Testverfahren, White-Box-Testverfahren und erfahrungsbasierten Testverfahren erklären können
- Black-Box-Testverfahren zur Ermittlung von Testfällen erklären, anwenden und voneinander abgrenzen können
- Die Äquivalenzklassenbildung und die Grenzwertanalyse kennen und auf einfache Beispiele anwenden können
- Den zustandsbasierten Test und den Entscheidungstabellentest kennen und auf einfache Beispiele anwenden können
- Den anwendungsfallbasierten Test erklären können und weitere Black-Box-Testverfahren nennen können

# Lernziele für den Abschnitt Testverfahren

(nach Certified Tester Foundation Level Syllabus,  
deutschsprachige Ausgabe, Version 2018)



- **4.1 Kategorien von Testverfahren**

- FL-4.1.1 (K2) Die Eigenschaften, Gemeinsamkeiten und Unterschiede zwischen Black-Box-Testverfahren, White-Box-Testverfahren und erfahrungsbasierten Testverfahren erklären können.

- **4.2 Black-Box-Testverfahren**

- FL-4.2.1 (K3) Die Äquivalenzklassenbildung anwenden können, um Testfälle aus vorgegebenen Anforderungen abzuleiten.
- FL-4.2.2 (K3) Die Grenzwertanalyse anwenden können, um Testfälle aus vorgegebenen Anforderungen abzuleiten.
- FL-4.2.3 (K3) Entscheidungstabellentests anwenden können, um Testfälle aus vorgegebenen Anforderungen abzuleiten.
- FL-4.2.4 (K3) Zustandsbasierte Tests anwenden können, um Testfälle aus vorgegebenen Anforderungen abzuleiten.
- FL-4.2.5 (K2) Erklären können, wie man Testfälle aus einem Anwendungsfall ableitet.

# Kategorien von Testverfahren



Photo: Fotolia / Dan Race



Source: <http://www.operation.de/knie/>

- **Black-Box-Testverfahren**

- Spezifikationen sind Grundlage für den Testentwurf, aber nicht der Programm Quelltext oder sein innerer Aufbau
- z.B. Äquivalenzklassenbildung und Grenzwertanalyse

- **White-Box-Testverfahren**

- Programm Quelltext liegt vor und ist Grundlage für den Testentwurf
- z.B. Entscheidungsüberdeckung oder Anweisungsüberdeckung

- **Erfahrungsbasiertes Testen**



Black-box vs. White-box

**4.1**  
**Dynamischer**  
**Test**  
**Black-Box**

**Dynamischer Test – Grundlagen**

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

Zustandsbasierter Test

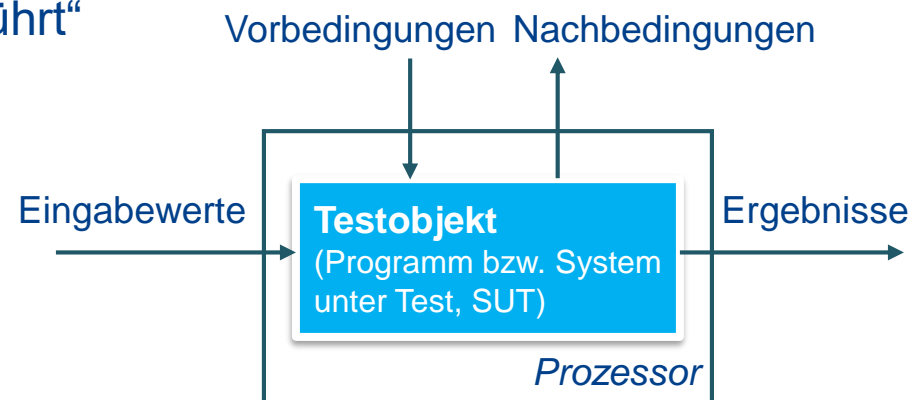
Entscheidungstabellentest

Anwendungsfallbasierter Test

Weitere Black-Box-Testverfahren

# Statischer Test vs. dynamischer Test

- Programme: statische Beschreibungen von dynamischen Abläufen (Berechnungen)
- Statische Tests prüfen die Testobjekte an sich
  - Artefakte des Entwicklungsprozesses, z.B. informelle Texte, Modelle, formale Texte, Programmcode, ...
- Dynamische Tests prüfen die durch Interpretation einer Beschreibung (Testobjekt) resultierenden Abläufe
- Das Testobjekt wird im dynamischen Test also auf einem Prozessor „ausgeführt“
  - Bereitstellen von Eingabewerten
  - Beobachten der Ergebnisse





# Dynamischer Test – Begriffe (1 von 6)

## **Dynamischer Test**

Prüfung des Testobjekts durch Ausführung auf einem Rechner

## **Testbasis**

Alle Informationen, die als Basis für die Testanalyse und den Testentwurf verwendet werden können. [nach TMap]

## **Testbedingung**

Ein Aspekt der Testbasis, der für die Erreichung bestimmter Testziele relevant ist.

## **Testziel**

Ein Grund oder Zweck für den Entwurf und die Ausführung von Tests.

(ISTQB Glossar V3.2)



## Dynamischer Test – Begriffe (2 von 6)

### **Testfall**

Eine Menge von Vorbedingungen, Eingaben, Aktionen (falls anwendbar), erwarteten Ergebnissen und Nachbedingungen, welche auf Basis von Testbedingungen entwickelt wurden. [Nach ISO 29119]

### **Abstrakter Testfall**

Ein Testfall ohne konkrete Werte für Eingaben und erwartete Ergebnisse.

### **Konkreter Testfall**

Ein Testfall mit konkreten Werten für Eingaben und vorausgesagte Ergebnisse.

### **Vorbedingung**

Der erforderliche Zustand des Testelements und seiner Umgebung vor der Ausführung eines Testfalls.

### **Nachbedingung**

Der erwartete Zustand eines Testelements und seiner Umgebung nach der Ausführung eines Testfalls.

(ISTQB Glossar V3.2)





## Dynamischer Test – Begriffe (3 von 6)

### Testsuite

Eine Menge von Testfällen oder Testabläufen, welche in einem bestimmten Testzyklus ausgeführt werden sollen.

### Testskript

Eine Abfolge von Anweisungen für die Durchführung eines Tests.

### Testausführungsplan

Ein Zeitplan für die Ausführung von Testsuiten innerhalb eines Testzyklus.  
(Anmerkung: Testskripte sind im Testausführungsplan mit ihrem Kontext und in der auszuführenden Reihenfolge festgelegt.)

### Testlauf

Die Ausführung eines oder mehrerer Testfälle mit einer bestimmten Version des Testobjekts.

(ISTQB Glossar V3.2)



# Dynamischer Test – Begriffe (3 von 6)

## Testfallspezifikation

Die Dokumentation von einem oder mehreren Testfällen. [ISO 29119]

## Testablaufspezifikation

Die Spezifikation eines oder mehrerer Testabläufe. [Nach ISO 29119]

## Testentwurfsspezifikation

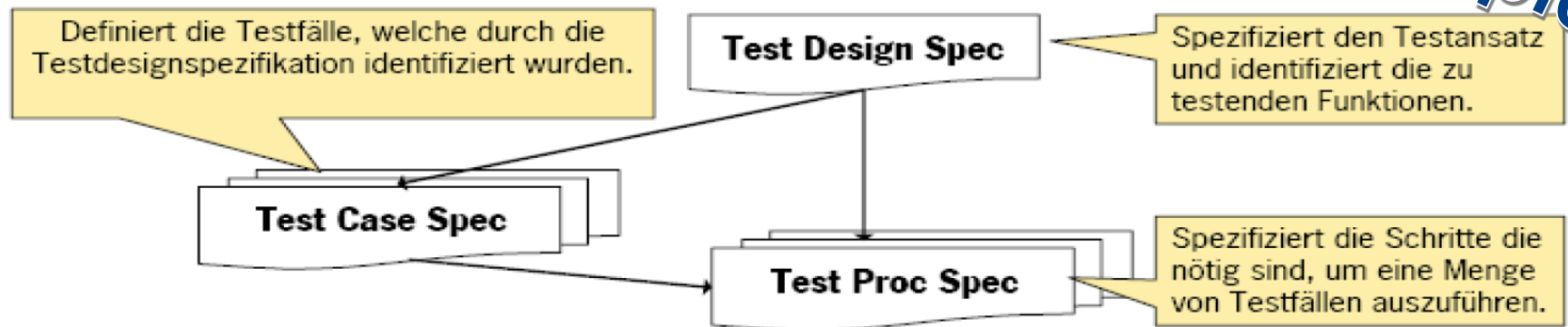
Die Spezifikation der zu testenden Features und ihrer zugehörigen Testbedingungen. [ISO 29119]

(ISTQB Glossar V3.2)

PORSCHE

## Systematische Testspezifikation

Beispiel:



Systematischer\_Test\_modellobasierter\_Steuergeräte.ppt  
Wiese, Feurte, Gorn, Zoller

Test modellbasiert entwickelter Steuergeräte

Erstelldatum: 30.09.2007  
Änderungsdatum: 04.12.2007  
Seite 29 von 32



# Dynamischer Test – Begriffe (4 von 6)

## Rückverfolgbarkeit

Der Grad, zu dem eine Beziehung zwischen zwei oder mehr Arbeitsergebnissen hergestellt werden kann. [Nach ISO 19506]

**Vertikale Rückverfolgbarkeit:** Die Rückverfolgung von Anforderungen durch die Ebenen der Entwicklungsdokumentation bis zu den Komponenten.

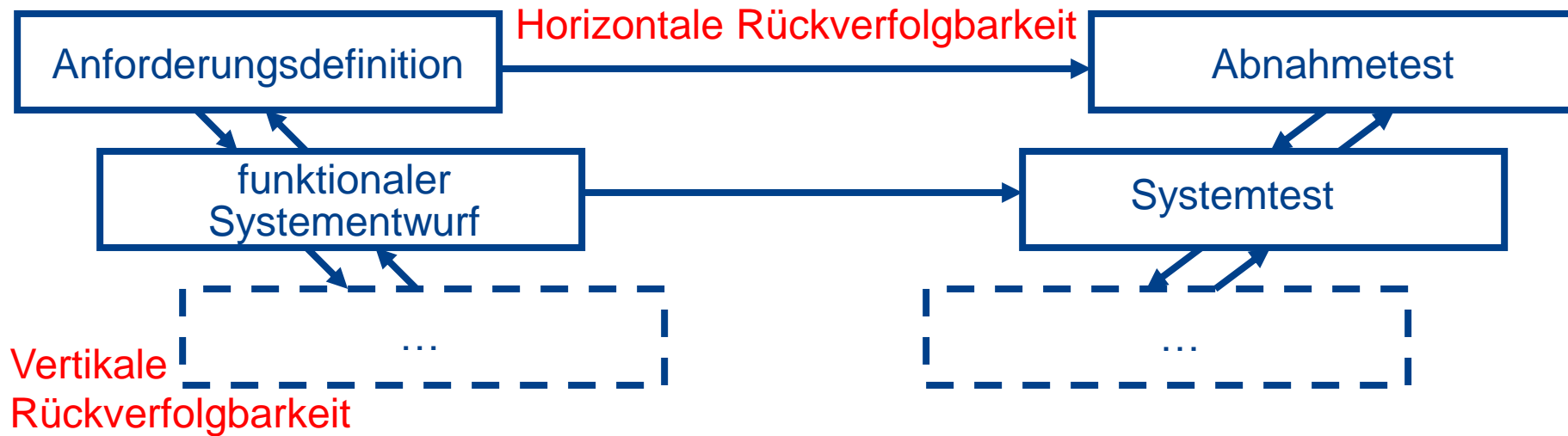
**Horizontale Rückverfolgbarkeit:** Das Verfolgen von Anforderungen einer Teststufe über die Ebenen der Testdokumentation (z.B. Testkonzept, Testentwurfsspezifikation, Testfallspezifikation, Testablaufspezifikation oder Testskripte).

**Rückverfolgbarkeitsmatrix:** Eine zweidimensionale Tabelle, die die gegenseitigen Beziehungen zweier Entitäten wie z.B. Anforderungen und Testfälle darstellt. Die Tabelle wird zur Bestimmung und Erreichung der Überdeckung verwendet, um von einer Entität zur anderen und zurück zu verfolgen, und um die Auswirkung von Änderungsvorschlägen zu bewerten.

Rückverfolgbarkeit erlaubt sowohl eine Analyse der Auswirkungen (impact analysis) aufgrund von geänderten Anforderungen als auch die Bestimmung einer Anforderungsüberdeckung, bezogen auf eine bestimmte Menge von Tests



# Rückverfolgbarkeit



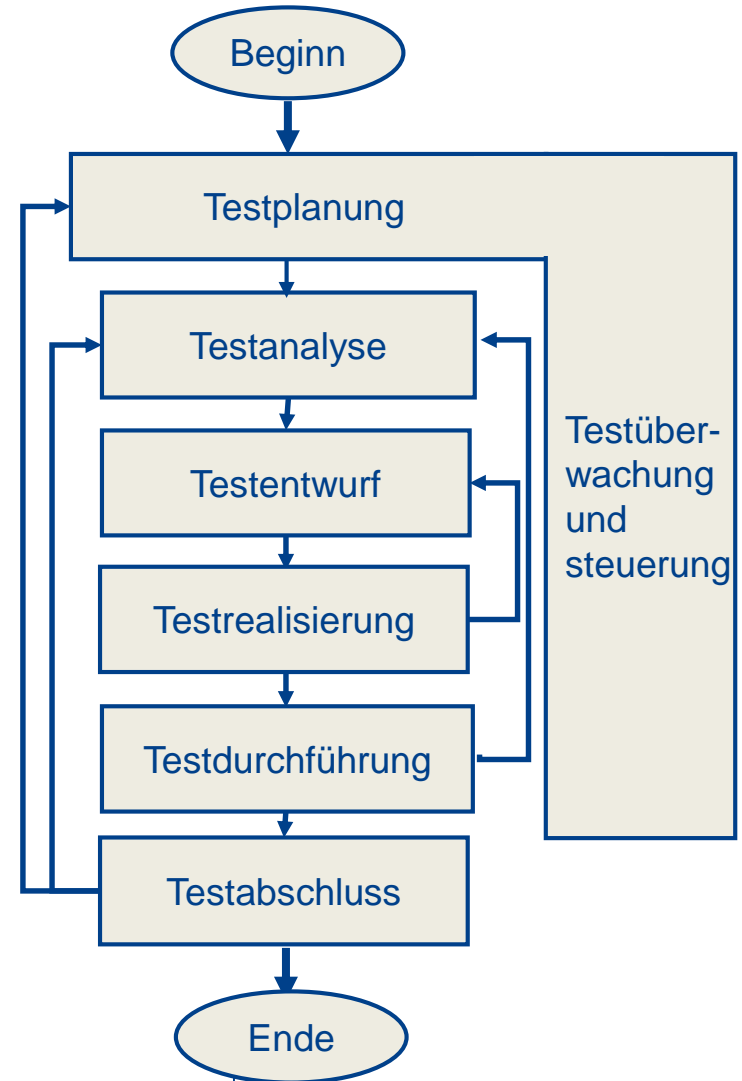
## Rückverfolgbarkeitsmatrix

### Testfälle

Anforderungen		TF1	TF2	TF3	TF4	TF5	TF6
	Req1	X	X				
	Req2			X			X
	Req3				X	X	
	Req4					X	X

# Dynamischer Test – Entwurf und Durchführung

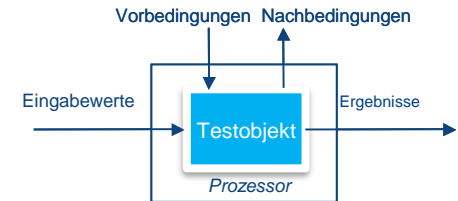
- siehe Testprozess nach CTFL Lehrplan 2018 in Kapitel 1
- Fokus auf
  - Testanalyse und -entwurf
    - Testbasis analysieren
    - Testbedingungen festlegen
    - Testfälle spezifizieren
    - Erwartetes Ergebnis VOR Durchführung spezifizieren
  - Testrealisierung und -durchführung
    - Reihenfolge der Tests definieren
    - Priorität, Regression, Abhängigkeiten... berücksichtigen
    - Tests manuell oder automatisch durchführen



# Dynamischer Test in unteren Teststufen

- **Ziele des dynamischen Tests**

- Nachweis der Erfüllung der Anforderungen durch das Testobjekt
- Nachweis durch stichprobenhafte Programmläufe
- Aufdeckung von eventuellen Abweichungen und Fehlerwirkungen
- mit wenig Aufwand möglichst viele Anforderungen überprüfen bzw. Fehler nachweisen



- **Voraussetzung:** Testobjekt muss ausführbar sein

- Einzelne Klasse, Modul/Komponente, Teilsystem in der Regel nicht allein lauffähig

- Bietet oft nur Operationen/Dienste für andere Softwareeinheiten an
- Fehlendes Hauptprogramm zur Koordination des Ablaufes
- Stützt sich oft auf Operationen/Dienste anderer Softwareeinheiten ab

- Für untere Teststufen Klassen-/Modultest, Komponententest und Integrationstest

- Testobjekt in entsprechenden Testrahmen einbetten



## Dynamischer Test – Begriffe (5 von 6)

### **Testrahmen** (test harness, test bed)

Eine Testumgebung, die aus den für die Testausführung benötigten Treibern und Platzhaltern besteht.

### **Testumgebung**

Eine Umgebung, die benötigt wird, um Tests auszuführen. Sie umfasst Hardware, Instrumentierung, Simulatoren, Softwarewerkzeuge und andere unterstützende Hilfsmittel. [nach IEEE 610]

### **Platzhalter** (stub)

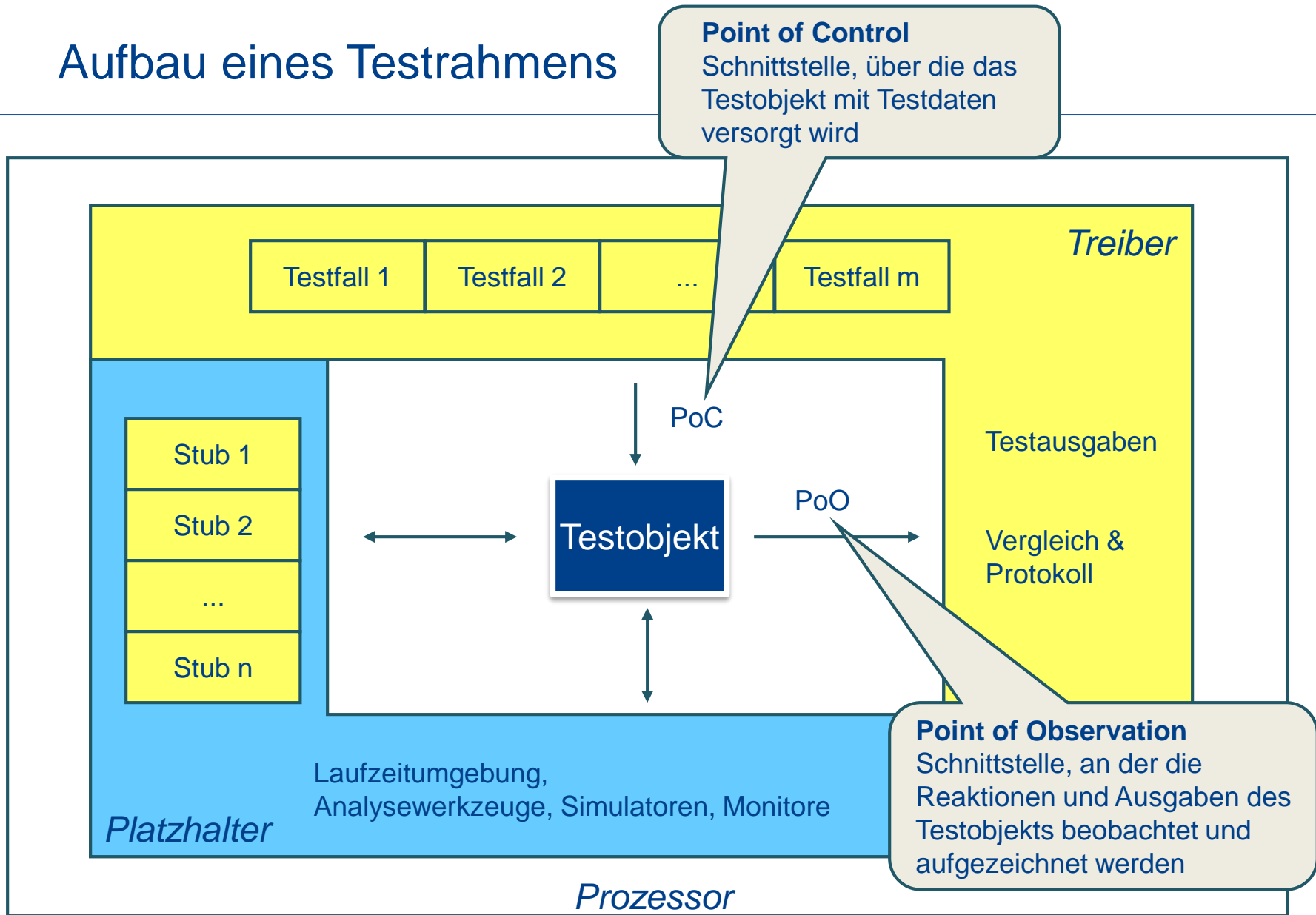
Eine rudimentäre oder spezielle Implementierung einer Softwarekomponente, die verwendet wird, um eine noch nicht implementierte Komponente zu ersetzen bzw. zu simulieren. [nach IEEE 610]

### **Treiber / Testtreiber** (driver)

Ein Testwerkzeug, das eine zu testende Komponente/ein System aufruft und/oder steuert. [nach TMap]

(ISTQB Glossar V3.2)

# Aufbau eines Testrahmens







# Dynamischer Test – Begriffe (6 von 6)

## Testverfahren

Eine Vorgehensweise, nach der Testfälle abgeleitet oder ausgewählt werden.

## Eingangskriterien

Die Menge an Bedingungen für den offiziellen Start einer bestimmten Aufgabe.

Beispiel: Eingangskriterium für Testentwurf:

Alle Systemanforderungen müssen gereviewt und freigegeben sein.

## Endekriterien

Die Menge an Bedingungen für den offiziellen Abschluss einer bestimmten Aufgabe.

Beispiel: Endekriterium für Testdurchführung:

Mindestens 80% des getesteten Quellcodes müssen durch Tests ausgeführt worden sein.

(ISTQB Glossar V3.2)

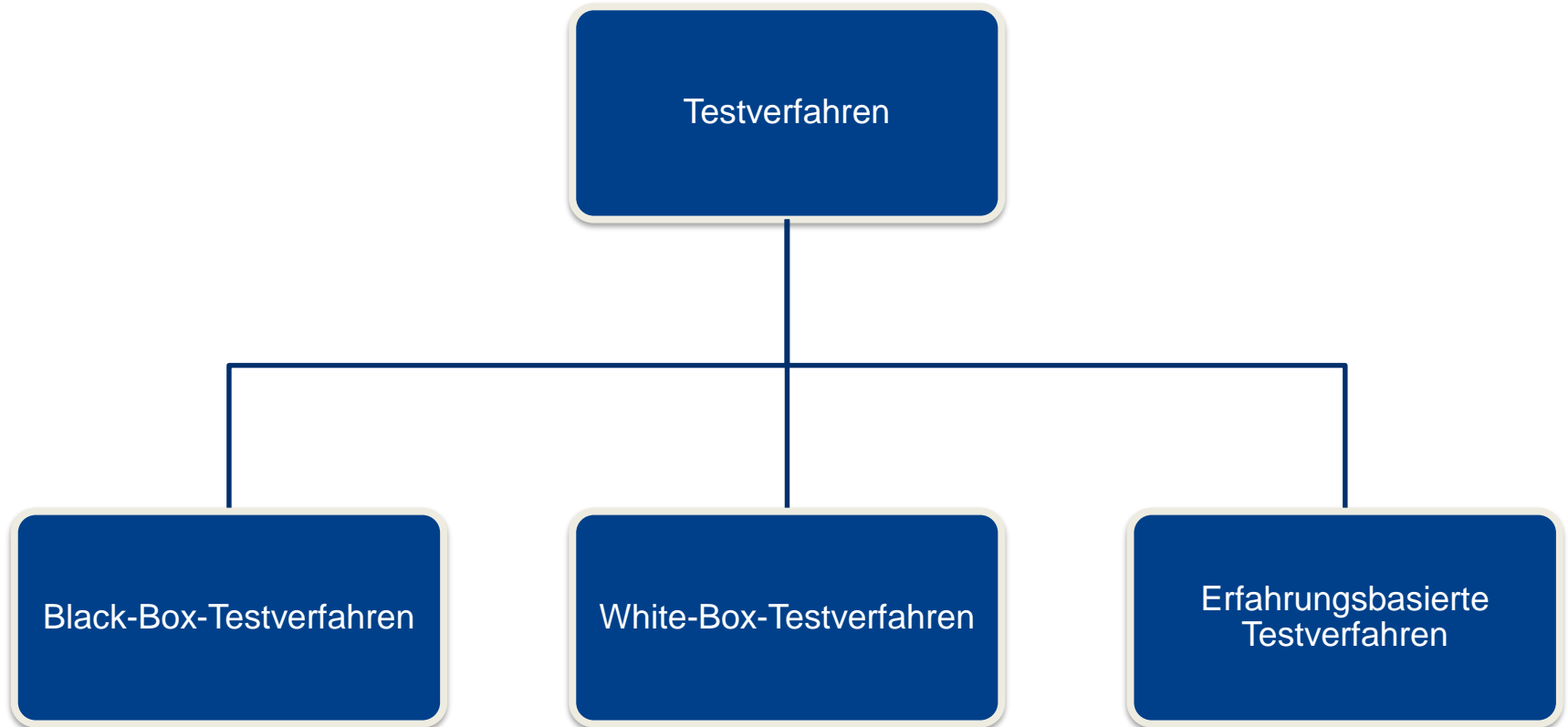
# Auswahl von Testverfahren (Test Techniques)



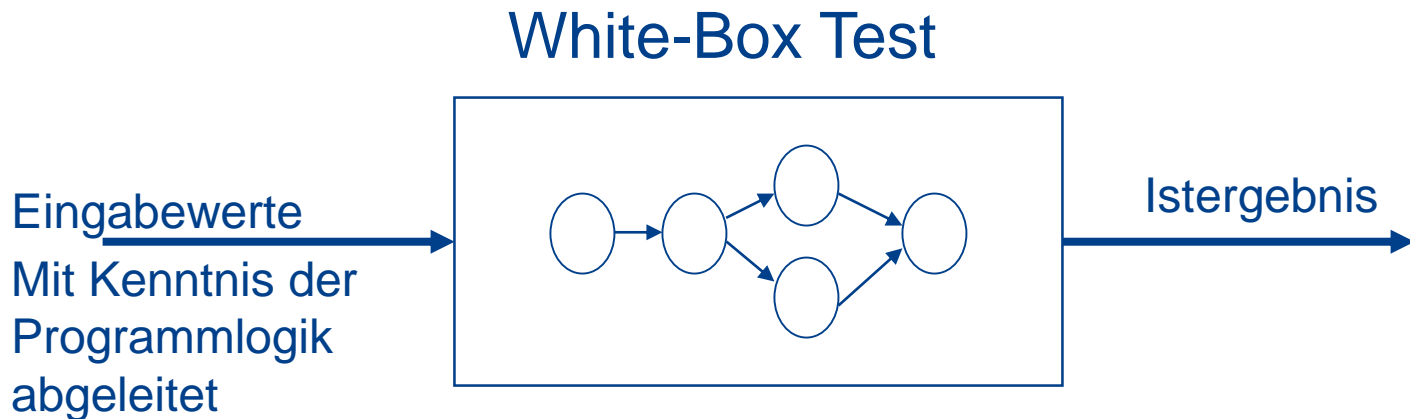
Die Zusammenhänge werden am Ende der Folien zu den Testverfahren im Detail erläutert.



# Testverfahren

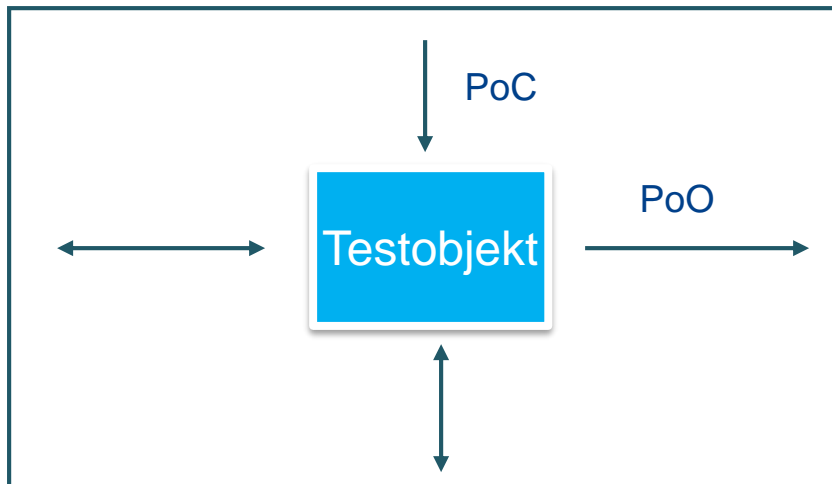


# Black-Box-Test vs. White-Box-Test (1 von 2)

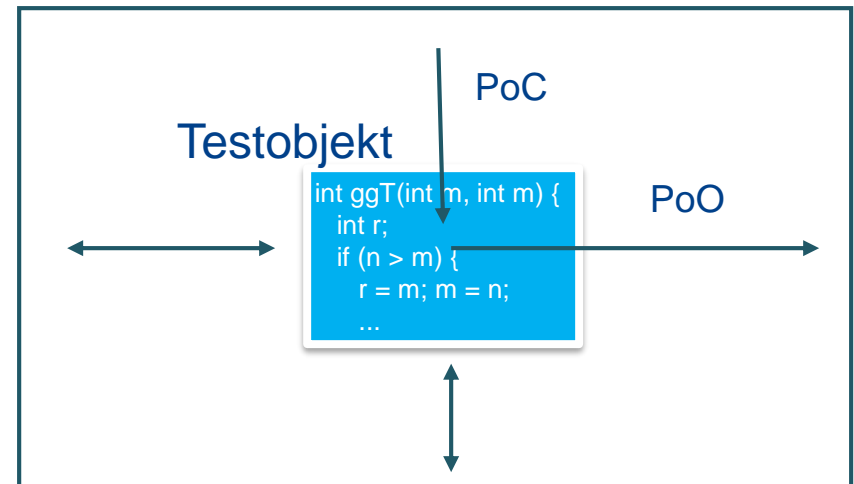


# Black-Box-Test vs. White-Box-Test (2 von 2)

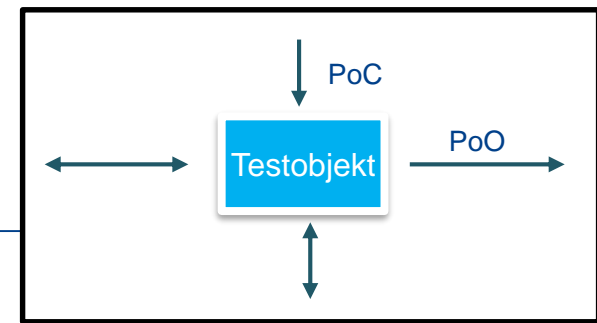
## Black-Box Test



## White-Box Test

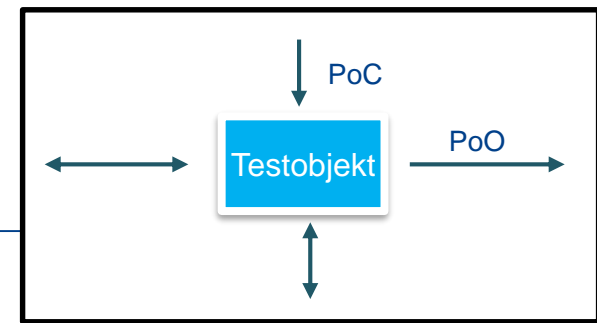


# Black-Box-Testverfahren (Testobjekt undurchsichtig)



- Keine Informationen über den Programmtext und den inneren Aufbau
- Steuert das Testobjekt nur durch die Wahl der Eingabetestdaten (PoC - Point of Control liegt außerhalb des Testobjektes)
- Beobachtet das Verhalten des Testobjekts von außen (PoO - Point of Observation liegt außerhalb des Testobjekts)
- Modelle bzw. Anforderungen an das Testobjekt, ob formal oder nicht formal, zur Spezifikation der Aufgabe, der Software oder ihrer Komponente heranziehen
- Von diesen Modellen auch systematisch Testfälle ableiten
- Synonyme:
  - spezifikationsorientierte Testverfahren
  - verhaltensgesteuerte Testverfahren

# White-Box-Testverfahren (Testobjekt durchsichtig)



- Testfälle basierend auf Programmstruktur des Testobjektes erzeugen
  - Ableiten von Testfällen aus Strukturinformation, beispielsweise aus Code
  - Messung der Überdeckung des Codes für vorhandene Testfälle
  - Erhöhung der Überdeckung durch systematische Ableitung von Testfällen
  - Analyse des inneren Ablaufs im Testobjekt während der Testdurchführung
    - Point of Observation liegt innerhalb des Testobjekts
  - Eingriff in den Ablauf im Testobjekt möglich
    - Point of Control kann innerhalb des Testobjekts liegen
    - notwendig, wenn erwartete Fehlbedienung über Schnittstelle nicht möglich
- Synonym: strukturorientierte Testverfahren

# Erfahrungsbasierte Testverfahren

---

- Wissen und Erfahrung von Menschen zur Ableitung der Testfälle nutzen
- Wissen von Testern, Entwicklern, Anwendern und Betroffenen über die Software, ihre Verwendung und ihre Umgebung
- Wissen über wahrscheinliche Fehler und ihre Verteilung



# Kapitel 4

## 4.1 Dynamischer Test Black-Box

Dynamischer Test – Grundlagen

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

Zustandsbasierter Test

Entscheidungstabellentest

Anwendungsfallbasierter Test

Weitere Black-Box-Testverfahren



# Spezifikationsorientierter Test – Begriffe

## **Funktionaler Test**

Testen, welches durchgeführt wird um die Erfüllung der funktionalen Anforderungen durch eine Komponente oder ein System zu bewerten. [ISO 24765]

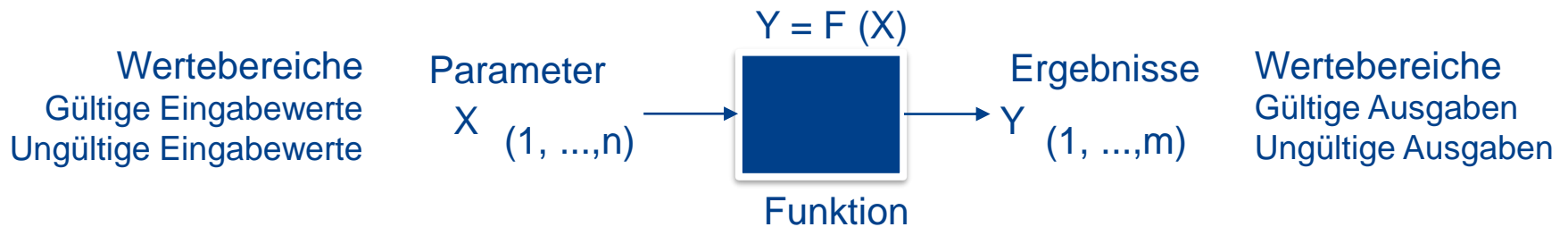
## **Funktionale Eignung**

Der Grad, zu dem eine Komponente oder ein System Funktionen zur Verfügung stellt, welche unter festgelegten Bedingungen explizit genannte und implizite Bedürfnisse erfüllen.

Untermerkmale der funktionalen Eignung nach ISO 25010 sind:  
Vollständigkeit, Korrektheit, Angemessenheit

(ISTQB Glossar V3.2)

# Spezifikationsorientierte Testfall- und Testdatenermittlung



- **Äquivalenzklassenbildung**

- Repräsentative Eingaben
- Gültige Dateneingaben
- Ungültige Dateneingaben
- Erreichen der gültigen Ausgaben

- **Grenzwertanalyse**

- Wertebereiche
- Wertebereichsgrenzen

- **Zustandsbasierter Test**

- Komplexe (innere) Zustände und Zustandsübergänge

- **Entscheidungstabellentest**

- Bedingungen und Aktionen

- **Anwendungsfallbasierter Test**

- Szenarien der Systemnutzung

# Kapitel 4

## 4.1 Dynamischer Test Black-Box

Dynamischer Test – Grundlagen

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

Zustandsbasierter Test

Entscheidungstabellentest

Anwendungsfallbasierter Test

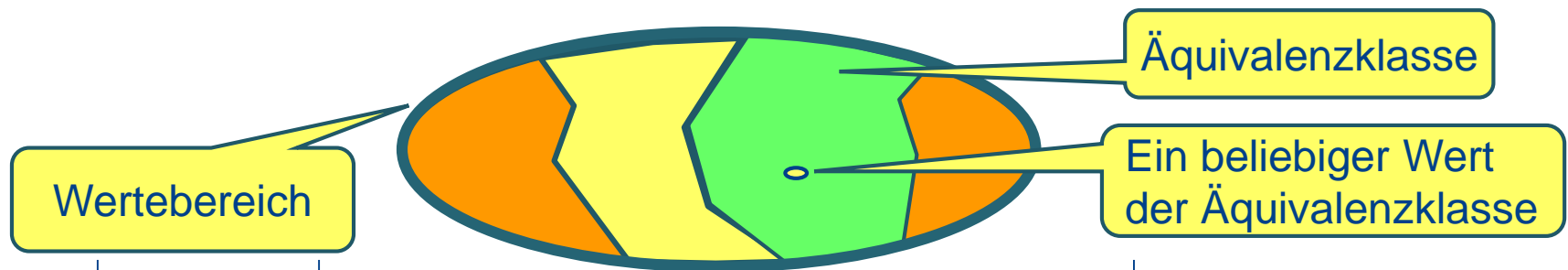
Weitere Black-Box-Testverfahren

# Grundidee der Äquivalenzklassenbildung

## Äquivalenzklassenbildung

Ein Black-Box-Testverfahren, bei dem die Testfälle im Hinblick auf die Ausführung von Äquivalenzklassen entworfen werden, wobei von jeder Äquivalenzklasse ein Repräsentant genutzt wird. [Nach ISO 29119]

- Wertebereiche der Ein-/Ausgaben in Äquivalenzklassen (ÄK) teilen
- Alle Werte einer Klasse zeigen äquivalentes Verhalten des Testobjekts
- Sinnvolle Stichprobe: nur ein Testwert pro ÄK (Repräsentant)
  - Wenn ein Wert der ÄK einen Fehler aufdeckt, wird erhofft, dass auch jeder andere Wert der ÄK diesen Fehler aufdeckt
  - Wenn ein Wert der ÄK keinen Fehler aufdeckt, wird erhofft, dass auch kein anderer Wert der ÄK einen Fehler aufdeckt



# Vorgehensweise

- Aufstellen der Wertebereiche aus der Spezifikation (Ein- und Ausgaben!)
- Äquivalenzklassenbildung für jede weitere Beschränkung
  - Falls eine Beschränkung einen Wertebereich spezifiziert:  
⇒ eine gültige und zwei ungültige ÄK
  - Falls eine Beschränkung eine minimale und maximale Anzahl von Werten spezifiziert:  
⇒ eine gültige und zwei ungültige ÄK
  - Falls eine Beschränkung eine Menge von Werten spezifiziert, die möglicherweise unterschiedlich behandelt werden:  
⇒ für jeden Wert dieser Menge eine eigene gültige ÄK und zusätzlich insgesamt eine ungültige ÄK
  - Falls eine Beschränkung eine Situation spezifiziert, die zwingend erfüllt sein muss:  
⇒ eine gültige und eine ungültige ÄK
  - Werden Werte einer ÄK vermutlich nicht gleichwertig behandelt:  
⇒ Aufspaltung der ÄK in kleinere ÄK

# Äquivalenzklassenbildung

## Beispiele (1 von 3)



- Falls eine Beschränkung einen Wertebereich spezifiziert:
  - eine gültige ÄK
  - zwei ungültige ÄK
    - Spezifikation: ganzzahlige Eingabewerte zwischen 1 und 100 sind möglich.
    - Wertebereich der Eingabe: ganzzahlig
    - Gültige Äquivalenzklasse:  $1 \leq x \leq 100$
    - Ungültige Äquivalenzklassen:  $x < 1$  sowie  $x > 100$
- Falls eine Beschränkung eine Anzahl von Werten spezifiziert:
  - eine gültige ÄK
  - zwei ungültige ÄK
    - Spezifikation: Ein Mitglied eines Sportvereins muss sich mindestens einer Sportart zuordnen. Jedes Mitglied kann an maximal drei Sportarten aktiv teilnehmen.
    - Gültige Äquivalenzklasse:  $1 \leq x \leq 3$  (1 bis 3 Sportarten)
    - Ungültige Äquivalenzklassen:  $x = 0$  (keine Sportart zugeordnet)  
 $x > 3$  (mehr als 3 Sportarten zugeordnet)

# Äquivalenzklassenbildung

## Beispiele (2 von 3)



- Falls eine Beschränkung eine Menge von Werten spezifiziert, die möglicherweise unterschiedlich behandelt werden:
  - für jeden Wert dieser Menge eine eigene gültige ÄK
  - zusätzlich insgesamt eine ungültige ÄK
- Spezifikation: Im Sportverein gibt es folgende Sportarten: Fußball, Hockey, Handball, Basketball und Volleyball.
- Gültige Äquivalenzklassen: Fußball, Hockey, Handball, Basketball, Volleyball
- Ungültige Äquivalenzklasse: alles andere, z.B. Badminton





# Äquivalenzklassenbildung

## Beispiele (3 von 3)

- Falls eine Beschränkung eine Situation spezifiziert, die zwingend erfüllt sein muss:
  - eine gültige ÄK
  - eine ungültige ÄK
- Spezifikation:
  - Jedes Mitglied im Sportverein erhält eine eindeutige Mitgliedsnummer.
  - Die Mitgliedsnummer beginnt mit dem ersten Buchstaben des Familiennamens.
- Gültige Äquivalenzklasse:
  - erstes Zeichen ist ein Buchstabe
- Ungültige Äquivalenzklasse:
  - erstes Zeichen ist kein Buchstabe (z.B. eine Ziffer oder ein Sonderzeichen)
  - Mitgliedsnummer besteht nur aus einem Buchstaben

# Anzahl der Testfälle minimieren: Heuristiken

- Testfälle aus allen Repräsentanten kombinieren und anschließend nach Häufigkeit sortieren (Benutzungsprofile)
  - Testfälle dann in dieser Reihenfolge priorisieren
  - Nur mit benutzungsrelevanten Testfällen testen
  - Testfälle bevorzugen, die Grenzwerte oder Kombinationen davon enthalten
- Jeden Repräsentanten einer Äquivalenzklasse mit jedem Repräsentanten jeder anderen Äquivalenzklasse testen
  - d.h. paarweise Kombination statt vollständiger Kombination
- Minimalkriterium: Mindestens ein Repräsentant jeder Äquivalenzklasse in mindestens einem Testfall
- Repräsentanten ungültiger Äquivalenzklassen nicht mit Repräsentanten anderer ungültiger Äquivalenzklassen kombinieren
  - Maximal eine ungültige ÄK pro Testfall; Fehlermaskierung möglich

# Äquivalenzklassenbildung: Endekriterium

- Endekriterium für den Test nach der Äquivalenzklassenbildung

$$\text{ÄK-Überdeckung} = \text{Anzahl getestete ÄK} / \text{Gesamtzahl ÄK}$$

- Beispiel:
  - Festgelegtes Endekriterium: ÄK-Überdeckung mindestens 80%
  - 18 Äquivalenzklassen aus der Testbasis ermittelt
  - 15 davon getestet
  - ÄK-Überdeckung =  $15 / 18 \approx 83,33 \%$
  - Endekriterium erfüllt

# Äquivalenzklassenbildung: Vor- und Nachteile

- Vorteile
  - Anzahl der Testfälle kleiner als bei unsystematischer Fehlersuche
  - Geeignet für Programme mit vielen verschiedenen Ein- und Ausgabebedingungen
- Nachteile
  - Betrachtet Bedingungen für einzelne Ein- oder Ausgabeparameter
  - Beachtung von Wechselwirkungen und Abhängigkeiten von Bedingungen sehr aufwändig
- Empfehlung
  - Zur Auswahl wirkungsvoller Testdaten:  
Kombination der ÄK-Bildung mit fehlerorientierten Verfahren,  
z.B. Grenzwertanalyse

# Kapitel 4

## 4.1 Dynamischer Test Black-Box

Dynamischer Test – Grundlagen

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

Zustandsbasierter Test

Entscheidungstabellentest

Anwendungsfallbasierter Test

Weitere Black-Box-Testverfahren

# Grenzwertanalyse (1 von 3)

## Grenzwertanalyse

Ein Black-Box-Testverfahren, bei dem die Testfälle unter Nutzung von Grenzwerten entworfen werden.[Nach ISO 29119] (Glossar ISTQB V3.2)

- Idee:  
In Verzweigungs- und Schleifenbedingungen gibt es oft Grenzbereiche, für welche die Bedingung gerade noch zutrifft (oder gerade nicht mehr)
  - Solche Fallunterscheidungen sind fehlerträchtig (*off by one*)
  - Testdaten, die solche Grenzwerte prüfen, decken Fehlerwirkungen mit höherer Wahrscheinlichkeit auf als Testdaten, die dies nicht tun
- Beste Erfolge bei Kombination mit anderen Verfahren
- Bei Kombination mit der Äquivalenzklassenbildung:
  - Grenzen der ÄK testen: pro ÄK kleinster und größter Wert
  - Jeder Rand einer ÄK muss in einer Testdatenkombination vorkommen

# Grenzwertanalyse (2 von 3)

- Grenzwerte von gültigen ÄK: „gültige“ Grenzwerte
- Grenzwerte von ungültigen ÄK: „ungültige“ Grenzwerte
- Die Grenzwerte der ÄK können um weitere zu testende Werte ergänzt werden, die die Wahrscheinlichkeit erhöhen, eine Fehlerwirkung zu entdecken
  - Werte rechts bzw. links neben den Grenzen  
(ungültige Werte, kleiner bzw. größer als Grenze)
- Mengenwertige Bereiche (z.B. bei Datenstrukturen, Beziehungen)
  - Kleinste und größte gültige Anzahl
  - Kleinste und größte ungültige Anzahl

# Grenzwertanalyse: Endekriterium

- Endekriterium für den Test nach der Grenzwertanalyse

$$\text{GW-Überdeckung} = \text{Anzahl getestete GW} / \text{Gesamtzahl GW}$$

- Beispiel:
  - Festgelegtes Endekriterium: GW-Überdeckung mindestens 90%
  - 20 Grenzwerte aus der Testbasis ermittelt
  - 19 davon getestet
  - $\text{ÄK-Überdeckung} = 19/20 = 95 \%$
  - Endekriterium erfüllt





# Grenzwertanalyse: Tipps (1 von 2)

- Grenzen des Eingabebereichs
  - Bereich: [-1.0;+1.0]
    - Testdaten: -1.001; -1.0; +1.0; +1.001 (-0.999; +0.999)
  - Bereich: ]-1.0;+1.0[;
    - Testdaten: -1.0; -0.999; +0.999; +1.0 (-1.001; +1.001)
- Grenzen der erlaubten Anzahl von Eingabewerten
  - Eingabedatei mit 1 bis 365 Sätzen
    - Testfälle 0, 1, 365, 366 (2, 364) Sätze
- Grenzen des Ausgabebereichs
  - Programm errechnet Beitrag, der zwischen 0,00 EUR und 600 EUR liegt
    - Testfälle: Für 0 EUR, 600 EUR und möglichst auch für Beiträge  $< 0$ ; (knapp  $> 0$ ); und für  $> 600$ ; (knapp  $< 600$ )
- Grenzen der erlaubten Anzahl von Ausgabewerten
  - Ausgabe von 1 bis 4 Daten
    - Testfälle: Für 0, 1, 4 und 5 (2, 3) Ausgabewerte



## Grenzwertanalyse: Tipps (2 von 2)

- Erstes und letztes Element bei geordneten Mengen beachten
  - z.B. sequentielle Datei, lineare Liste, Tabelle
- Bei komplexen Datenstrukturen leere Mengen testen
  - z.B. leere Liste, Null-Matrix
- Bei numerischen Berechnungen
  - eng zusammen und weit auseinander liegende Werte wählen

# Grenzwertanalyse: Vor- und Nachteile

- Vorteile

- An den Grenzen von Äquivalenzklassen sind häufiger Fehler zu finden als innerhalb dieser Klassen
- Die Grenzwertanalyse ist bei richtiger Anwendung eine der nützlichsten Methoden für den Testfallentwurf

Myers, Glenford J.:  
Methodisches Testen von Programmen  
Oldenbourg, 2001 (7. Auflage)

- Effiziente Kombination mit anderen Verfahren, die Freiheitsgrade in der Wahl der Testdaten lassen

- Nachteile

- Rezepte für die Auswahl von Testdaten schwierig anzugeben
- Bestimmung aller relevanten Grenzwerte schwierig
- Kreativität zur Findung erfolgreicher Testdaten gefordert
- Oft nicht effizient genug angewendet, da sie zu einfach erscheint

# Kapitel 4

## 4.1 Dynamischer Test Black-Box

Dynamischer Test – Grundlagen

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

**Zustandsbasierter Test**

Entscheidungstabellentest

Anwendungsfallbasierter Test

Weitere Black-Box-Testverfahren

# Zustandsbasierter Test

- Der bisherige Ablauf kann Einfluss auf das Systemverhalten haben
  - Ein endlicher Automat / Zustandsautomat (*finite state machine*) besteht aus einer endlichen Anzahl von internen Zuständen (Konfigurationen)
  - Der Zustand eines Systems beinhaltet implizit Informationen
    - aus den bisherigen Eingaben und
    - um die Reaktion des Systems auf noch folgende Eingaben zu bestimmen

H. Balzert: Lehrbuch der Softwaretechnik, Bd. I, Spektrum, 2002

- Das System nimmt unterschiedliche Zustände an
  - Zustandsänderungen durch Ereignisse (z.B. Funktionsaufrufe) ausgelöst
  - Bei den Zustandsänderungen können Aktionen durchzuführen sein
  - Spezielle Zustände sind:
    - Startzustand
    - Ein oder mehrere Endzustände



# Zustandsbasierter Test: Begriffe

## **Zustandsautomat**

Ein Berechnungsmodell, bestehend aus einer endlichen Anzahl von Zuständen und Zustandsübergängen, ggf. mit begleitenden Aktionen. [IEEE 610].

## **Zustandsübergang**

Ein Übergang zwischen zwei Zuständen einer Komponente oder eines Systems.

## **Zustandsdiagramm**

Ein Diagramm, das die Zustände beschreibt, die ein System oder eine Komponente annehmen kann, und die Ereignisse bzw. Umstände zeigt, die einen Zustandswechsel verursachen und/oder ergeben.

## **Zustandsübergangstabelle**

Eine Tabelle, die für jeden Zustand in Verbindung mit jedem möglichen Ereignis die resultierenden Übergänge darstellt. Das können sowohl gültige als auch ungültige Übergänge sein.

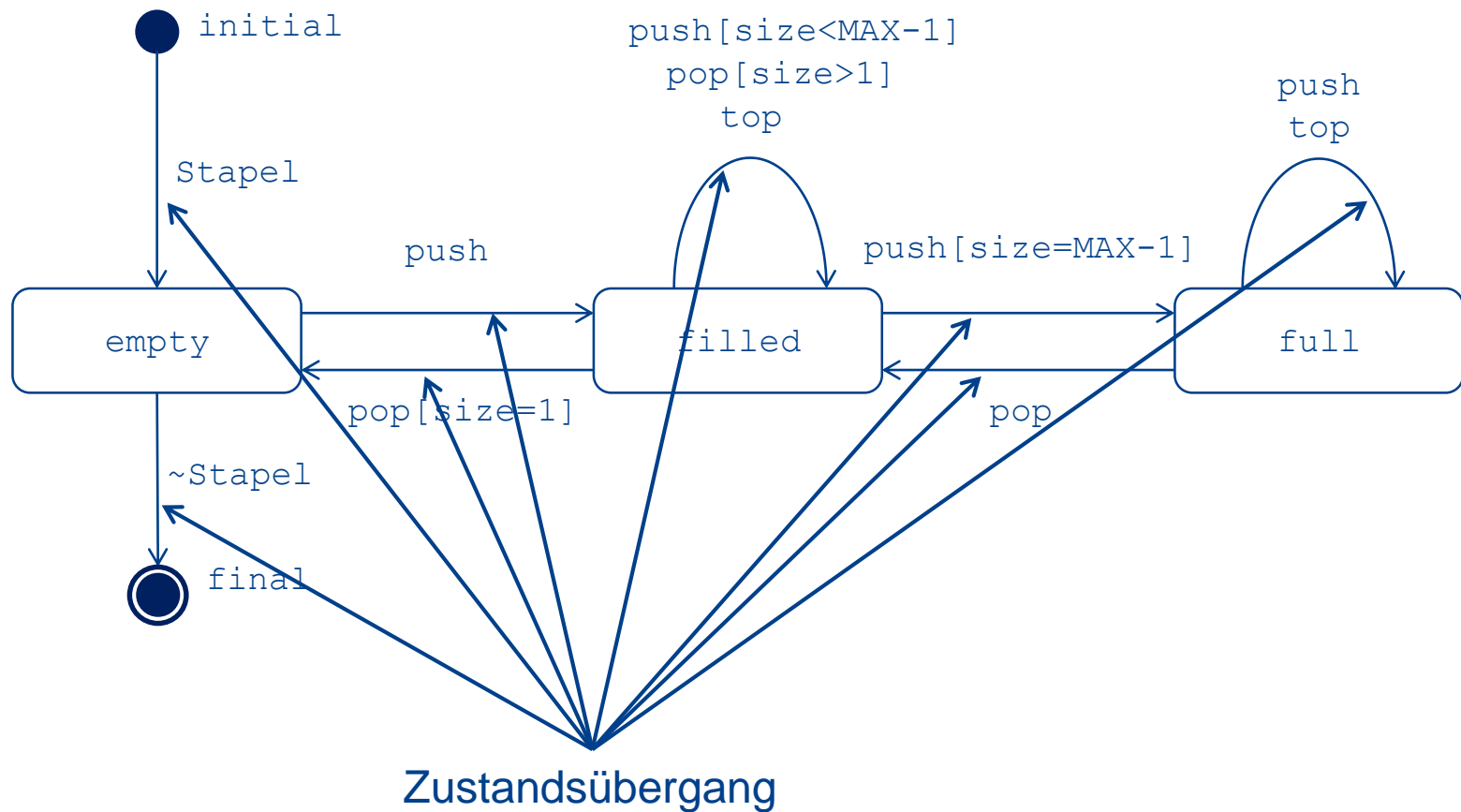
## **Zustandsbasierter Test**

Ein Black-Box-Testverfahren, bei dem Testfälle aus Zustandsdiagrammen oder Zustandstabellen abgeleitet werden, um zu bewerten ob das Testelement gültige Zustandsübergänge erfolgreich ausführt und ungültige Übergänge verhindert.



# Zustandsbasierter Test: Begriffe

## Zustandsdiagramm





# Zustandsbasierter Test: Begriffe

## Zustandsübergangstabelle

Zustand	Ereignis	Wächter- bedingung	Effekt	Folgezustand
initial	Stapel	-	Stapel-Objekt wird erzeugt, size = 0	empty
empty	push	-	Objekt wird auf Stapel gelegt, size = 1	filled
filled	push	size < MAX-1	Objekt wird auf Stapel gelegt, size = size +1	filled
filled	push	size = MAX-1	Objekt wird auf Stapel gelegt, size = size +1, size = MAX	full
full	push	-	-	full
...	...	...	...	...





# Beispiel zur Zustandsmodellierung: Stapel (Stack)

## Klasse Stapel

### Zustandserhaltende Operationen

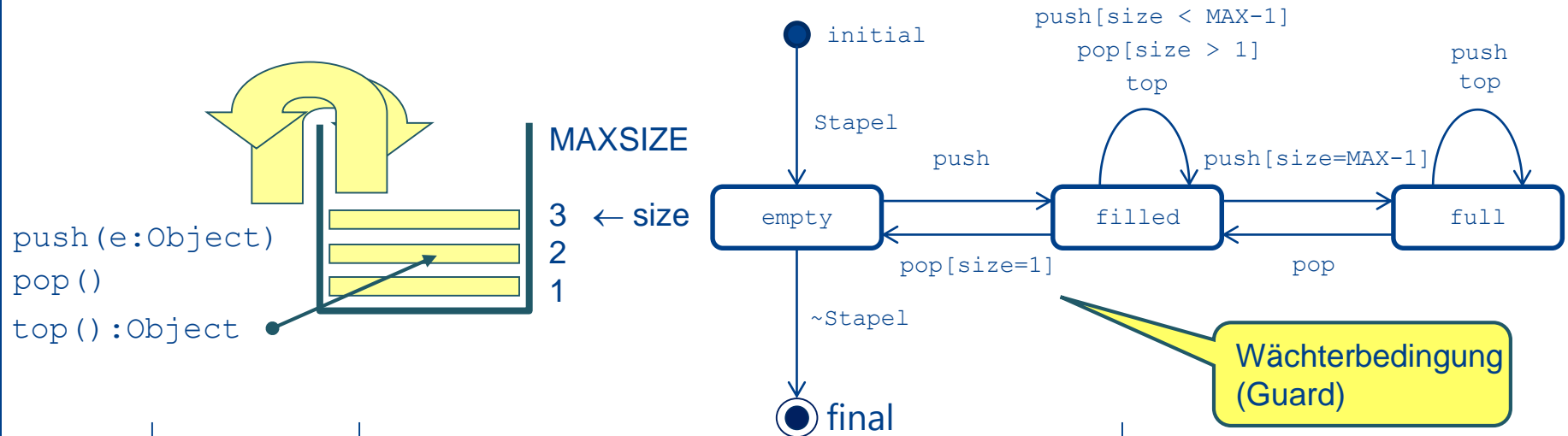
```
size():integer; // Anzahl gestapelter Elemente  
MAX():integer; // Maximale Anzahl  
top():Object; // Zeiger auf oberstes Element
```

### Zustandsverändernde Operationen

```
Stapel(Max:integer); // Konstruktor  
~Stapel(); // Destruktor  
push(element:Object); // Stapelt Element  
pop(); // Entfernt oberstes Element
```

### Drei Zustände (bei MAX() ≥ 2):

```
empty: size() = 0;  
filled: 0 < size() < MAX();  
full: size() = MAX();
```



# Ziele des zustandsbasierten Tests

---

- Nachweis, dass sich das Testobjekt konform zum Zustandsdiagramm verhält (Zustands-Konformanztest)
- Zusätzlich Test unter nicht-konformanten Benutzungen (Zustands-Robustheitstest)

# Arbeitsschritte des zustandsbasierten Tests

1. Fokussierung auf das Zustandsdiagramm
2. Prüfung auf Vollständigkeit
3. Ableiten des Übergangsbaumes für den Zustands-Konformanztest
4. Erweitern des Übergangsbaumes für den Zustands-Robustheitstest
5. Generieren der Botschaftssequenzen und Ergänzen der Botschaftsparameter
6. Ausführen der Tests und Überdeckungsmessung

# 1. Fokussierung auf das Zustandsdiagramm

## Drei Zustände:

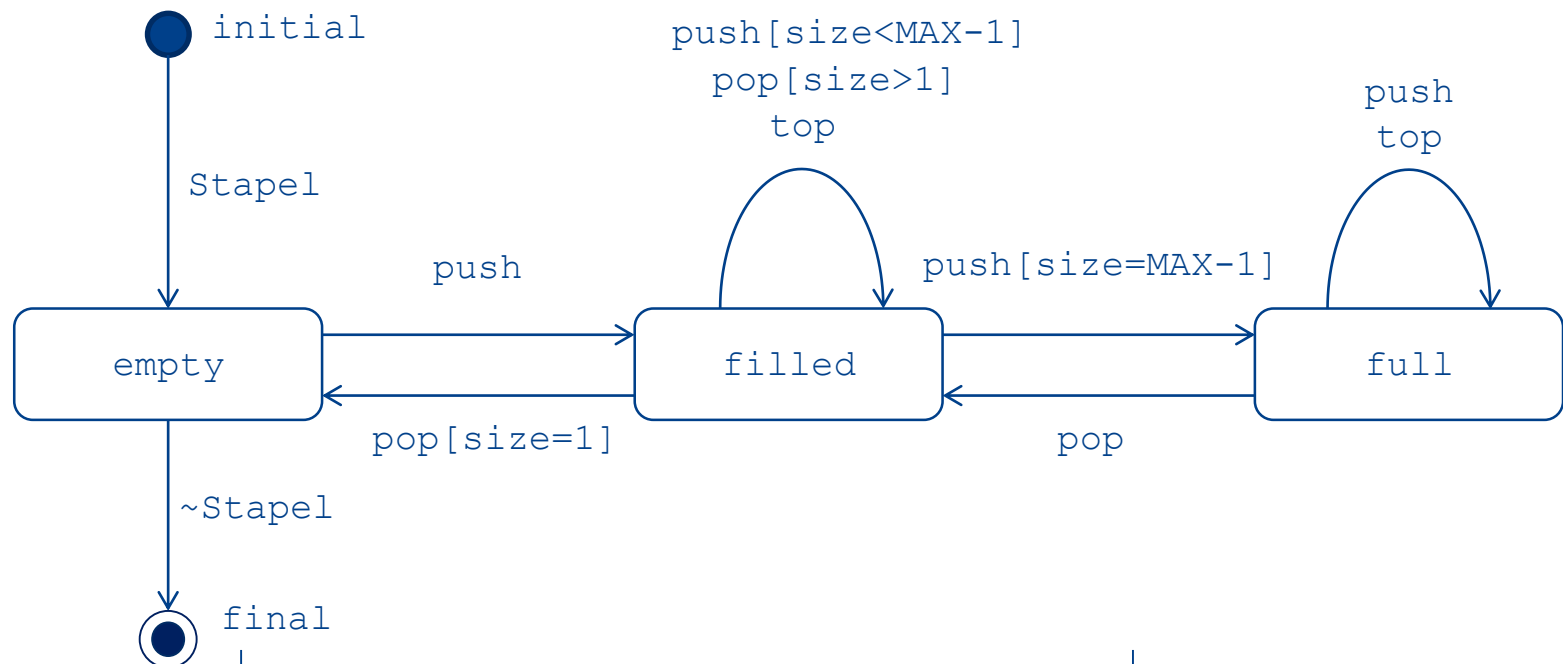
empty: `size() = 0;`  
filled: `0 < size() < MAX();`  
full: `size() = MAX();`

## Zwei Pseudo-Zustände:

initial: Vor Erzeugung;  
final: Nach Zerstörung

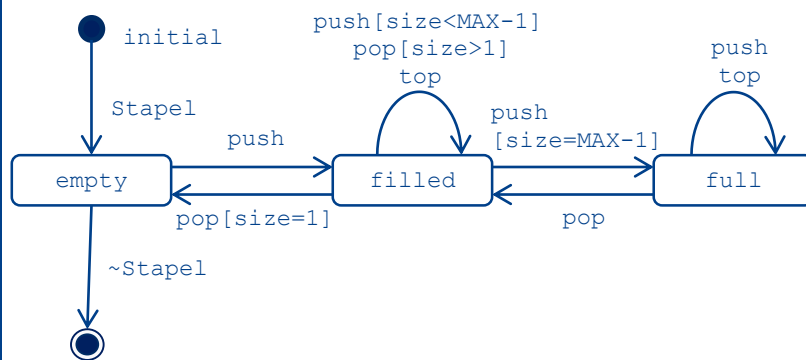
## Acht Zustandsübergänge:

initial  $\rightarrow$  empty; empty  $\rightarrow$  final  
empty  $\rightarrow$  filled; filled  $\rightarrow$  empty (Zyklus!)  
filled  $\rightarrow$  full; full  $\rightarrow$  filled (Zyklus!)  
filled  $\rightarrow$  filled; full  $\rightarrow$  full (Zyklen!)



## 2. Prüfung auf Vollständigkeit

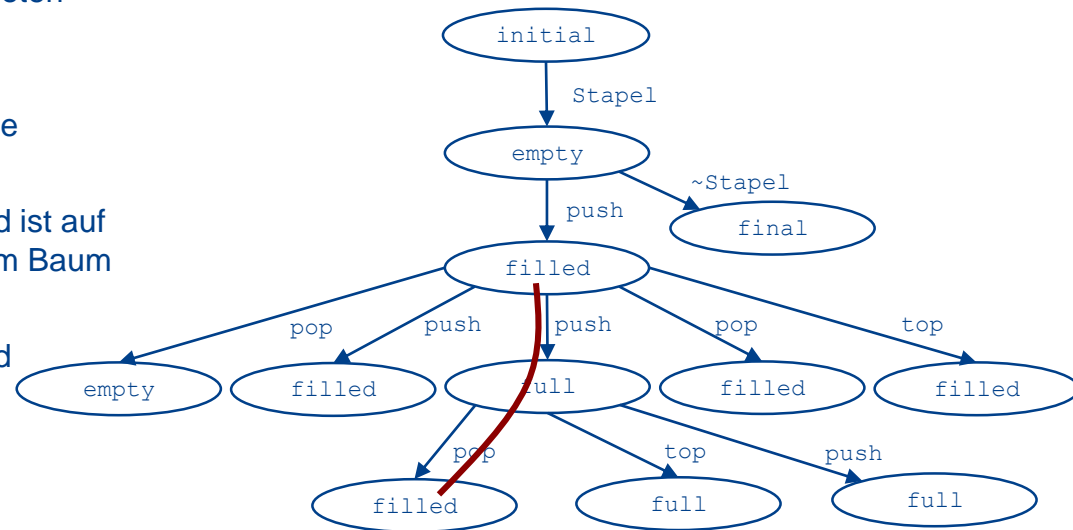
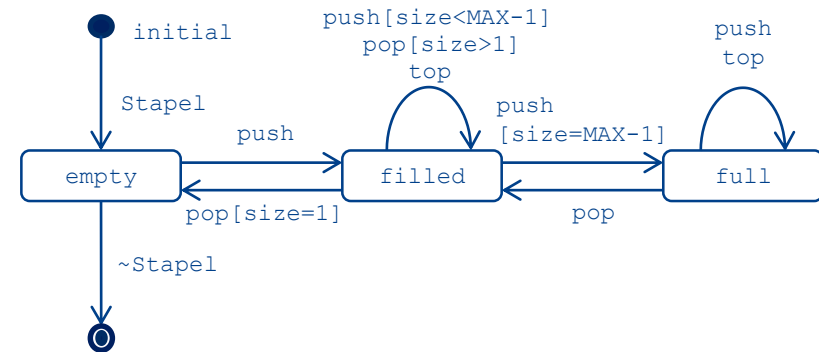
- Zustandsdiagramm hinsichtlich der Vollständigkeit untersuchen
  - Ggf. Zustandsübergangstabelle anlegen
  - Ggf. auch die Wächterbedingungen bez. eines Ereignisses auf Vollständigkeit und Konsistenz prüfen
  - Nicht spezifizierte Zustands/Ereignis-Paare hinterfragen



Zustand Ereignis	initial	empty	filled	full
Stapel()	empty	N/A	N/A	N/A
~Stapel()	N/A	final	?	?
push()	N/A	filled	filled, full	full
pop()	N/A	?	empty, filled	filled
top()	N/A	?	filled	full

# 3. Aufbau des Übergangsbaumes: Zustands-Konformanztest

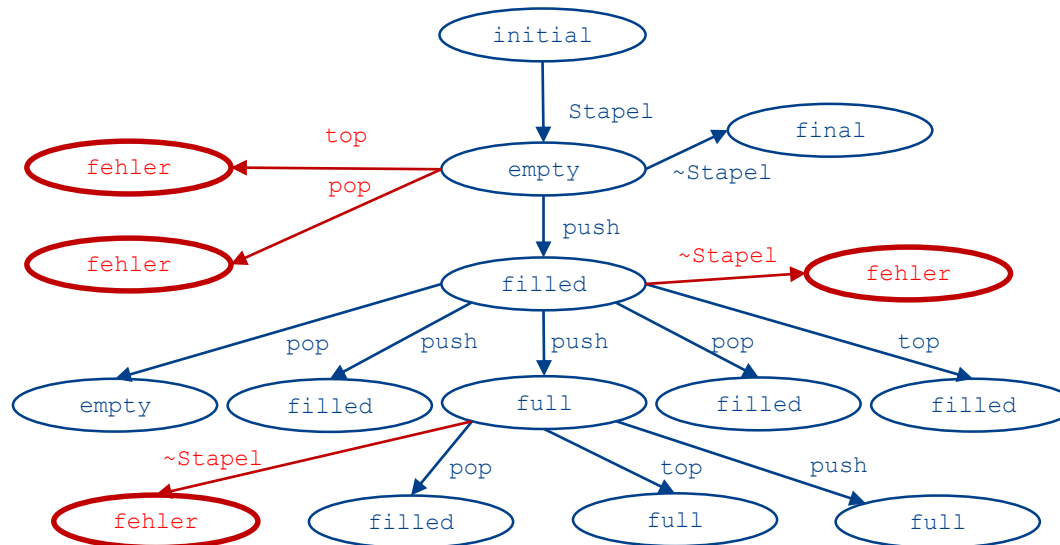
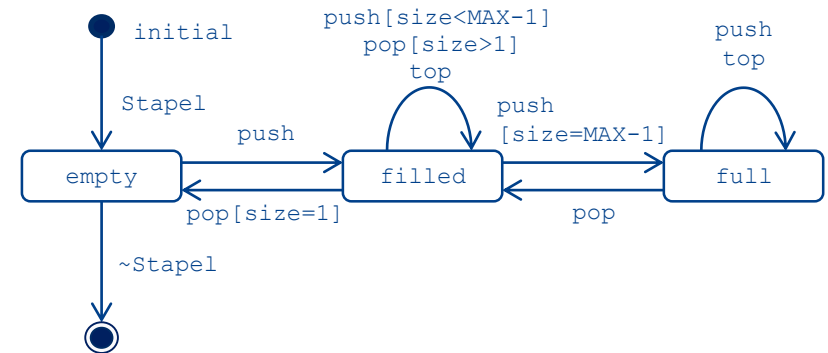
1. Der Anfangszustand wird die Wurzel des Baumes.
2. Für jeden möglichen Übergang vom Anfangszustand zu einem Folgezustand im Zustandsdiagramm erhält der Übergangsbaum von der Wurzel aus einen Zweig zu einem Knoten, der den Nachfolgezustand repräsentiert. Am Zweig wird das Ereignis (Operation) und ggf. die Wächterbedingung notiert. Ein Knoten ohne Übergang zu einem Folgeknoten heißt Blatt.
3. Der letzte Schritt wird für jedes Blatt des Übergangsbaums so lange wiederholt, bis eine der beiden Endbedingungen eintritt:
  - Der dem Blatt entsprechende Zustand ist auf einer höheren Ebene bereits einmal im Baum enthalten.
  - Der dem Blatt entsprechende Zustand ist ein Endzustand und hat somit keine weiteren Übergänge, die zu berücksichtigen wären.



(Wächterbedingungen hier nicht dargestellt)

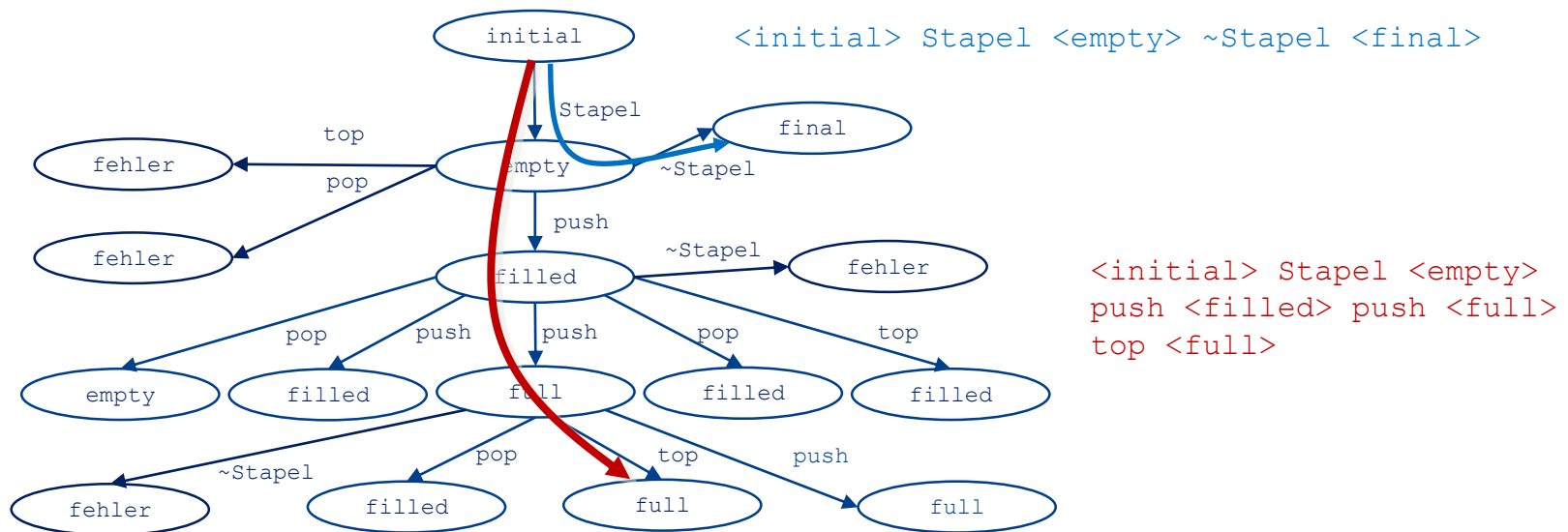
# 4. Erweitern des Übergangsbaumes: Zustands-Robustheitstest

- Robustheit unter spezifikationsverletzenden Benutzungen prüfen
- Für Botschaften, für die aus dem betrachteten Knoten kein Übergang spezifiziert ist, den Übergangsbaum um einen neuen Fehler-Zustand erweitern



## 5. Generieren der Testfälle (1 von 2)

- Pfade von der Wurzel zu Blättern im erweiterten Übergangsbaum als Funktions-Sequenzen auffassen
- Stimulierung des Testobjekts mit den entsprechenden Funktionsaufrufen deckt alle Zustände und Zustandsübergänge im Zustandsdiagramm ab
- Ergänzen der Parameter!





## 5. Generieren der Testfälle (2 von 2)

- Pfade von der Wurzel zu Blättern im erweiterten Übergangsbaum als Funktions-Sequenzen auffassen
- Stimulierung des Testobjekts mit den entsprechenden Funktionsaufrufen deckt alle Zustände und Zustandsübergänge im Zustandsdiagramm ab
- Parameter ergänzen! Wächterbedingungen beachten!

### Zustands-Konformanztest:

```
K1 = <initial> new Stapel() <empty> ~Stapel() <final>
K2 = <initial> new Stapel() <empty> push() <filled> pop() <empty>
K3 = <initial> new Stapel() <empty> push() <filled> push() <filled>
K4 = <initial> new Stapel() <empty> push() <filled> push() <filled>
...
K8 = <initial> new Stapel() <empty> push() <filled> push() <full> top() <full>
```

Wächterbedingung:  
size() > 1 !!

push() <filled> pop() <filled>

### Zustands-Robustheitstest:

```
R1 = <initial> new Stapel() <empty> pop() <fehler>
R2 = <initial> new Stapel() <empty> top() <fehler>
R3 = <initial> new Stapel() <empty> push() <filled> ~Stapel() <fehler>
R4 = <initial> new Stapel() <empty> push() <filled> push() <full> ~Stapel() <fehler>
```

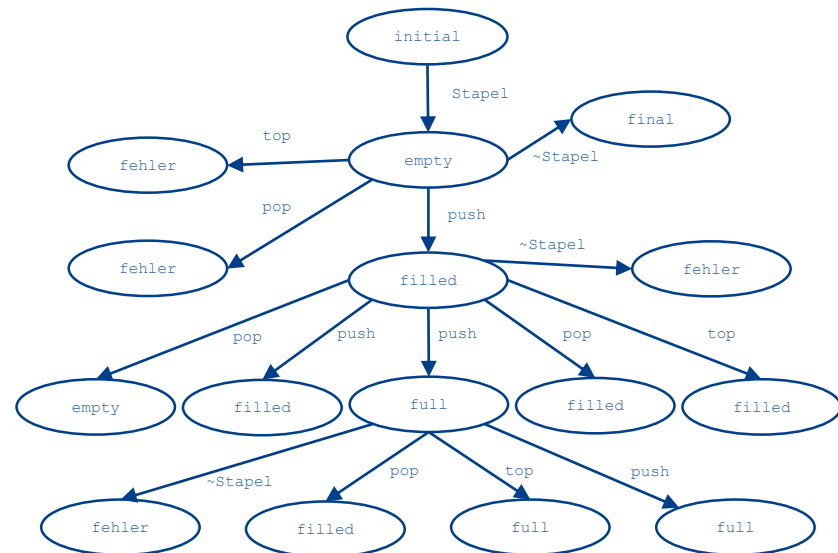
# Zustandsbasierte Testfälle

- Ein vollständiger zustandsbasierter Testfall umfasst folgende Informationen
  - Anfangszustand des Testobjektes (Komponente oder System)
  - Eingaben für das Testobjekt
  - Erwartete Ausgaben bzw. das erwartete Verhalten
  - Erwarteter Endzustand
- Ferner sind für jeden im Testfall erwarteten Zustandsübergang folgende Aspekte festzulegen
  - Zustand vor dem Übergang
  - Auslösendes Ereignis, das den Übergang bewirkt
  - Erwartete Reaktion, ausgelöst durch den Übergang
  - Nächster erwarteter Zustand

## 6. Ausführen der Tests

- Testfälle bzw. Botschaftsfolgen in ein Testskript verkapseln
- Unter Benutzung eines Testtreibers ausführen
- Zustände über zustandserhaltende Operationen ermitteln und protokollieren

```
K3' = //<initial>  
      Stapel OUT = new Stapel(5)  
      //<empty>  
  
      OUT.push(new Object())  
      //<filled>  
  
      OUT.push(new Object())  
      //<filled>  
  
      if (OUT.size() != 2) then  
          throw  
              WrongStateException;
```



## 7. Zustandsbasierter Test: Endekriterien und Überdeckungsmessung

- Minimalkriterium: Jeder Zustand wurde mindestens einmal eingenommen (Endekriterium: Z-Überdeckung = 100%)

$$\text{Z-Überdeckung} = \text{Anzahl getestete Z} / \text{Gesamtzahl Z}$$

- Weitere mögliche Kriterien:
  - Jeder Zustandsübergang wurde mindestens einmal ausgeführt (Endekriterium: ZÜ-Überdeckung = 100%)

$$\text{ZÜ-Überdeckung} = \text{Anzahl getestete ZÜ} / \text{Gesamtzahl ZÜ}$$

- Alle spezifikationsverletzenden Zustandsübergänge wurden angeregt
  - Jede Aktion (Funktion) wurde mindestens einmal ausgeführt
- Bei hoch kritischen Anwendungen
  - Alle Zustandsübergänge und alle Zyklen im Zustandsdiagramm
  - Alle Zustandsübergänge in jeder beliebigen Reihenfolge mit allen möglichen Zuständen, auch mehrfach hintereinander

# Tipps für den zustandsbasierten Test

- Spezifikation ergibt eine hohe Anzahl von Zuständen und Übergängen
  - dann auf erhöhten Testaufwand hinweisen und auf Vereinfachung dringen
- Bei der Spezifikation darauf achten, dass
  - die unterschiedlichen Zustände leicht zu ermitteln sind und
  - sich nicht aus einer vielfältigen Kombination von Werten von unterschiedlichen Variablen ergeben
- Ggf. Werkzeuge zum Model-Checking verwenden
  - Erreichbarkeit von Zuständen
  - Bei interagierenden Testobjekten: Livelock, Deadlock, ...

# Bewertung des zustandsbasierten Tests (1 von 2)

- Zustand besteht aus mehreren unterschiedlichen Werten von Variablen:
  - aufgespannter Zustandsraum oft SEHR komplex
  - Überprüfung und Bewertung der Testfälle kann sehr aufwändig werden
- Zustandsbasierter Test, wenn
  - Zustände eine Rolle spielen
  - Funktionalität durch den jeweiligen Zustand unterschiedlich beeinflusst wird
- Hinweis:
  - Andere Testverfahren sind unabhängig vom Zustand

# Bewertung des zustandsbasierten Tests (2 von 2)

- Besonders geeignet zum Test objektorientierter Systeme
  - Objekte können unterschiedliche Zustände annehmen
  - Die jeweiligen Methoden zur Manipulation der Objekte müssen dann entsprechend auf die unterschiedlichen Zustände reagieren
  - Beim objektorientierten Testen hat der zustandsbasierte Test deshalb eine herausgehobene Bedeutung, da er diesen speziellen Aspekt der Objektorientierung berücksichtigt

# Kapitel 4

## 4.1 Dynamischer Test Black-Box

Dynamischer Test – Grundlagen

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

Zustandsbasierter Test

Entscheidungstabellentest

Anwendungsfallbasierter Test

Weitere Black-Box-Testverfahren



# Entscheidungstabellentest

## Entscheidungstabellentest

Ein Black-Box-Testverfahren, bei dem Testfälle im Hinblick auf die Ausführung von Kombinationen der Bedingungen einer Entscheidungstabelle entworfen werden. [Egler63] (ISTQB Glossar V3.2)

- Voraussetzung: Anforderungen mit logischen Bedingungen und komplexen Regeln
- Vorgehen:
  - Spezifikation untersuchen
  - Eingabebedingungen und Aktionen des Systems ermitteln
  - müssen entweder wahr oder falsch sein können
- Entscheidungstabelle für alle Eingabebedingungen und die resultierenden Aktionen
- Jede Spalte der Tabelle
  - definiert eindeutige Kombination der Bedingungen (Regel)
  - definiert mit dieser Regel verbundene Aktionen
- Üblicherweise verwendete Standardüberdeckung:
  - Wenigstens ein Testfall pro Spalte

# Entscheidungstabellentest

## Entscheidungstabellentest – Die vier Quadranten einer Entscheidungstabelle

Bedingungen	Regeln
Aktionen	Aktionszeiger

- Bedingungen:
  - Mögliche Zustände von Objekten
- Regeln:
  - Kombinationen von Bedingungswerten
- Aktionen:
  - Aktivitäten, die abhängig von den Regeln auszuführen sind
- Aktionszeiger:
  - Belegungen der Bedingungen mit Aktionen



# Entscheidungstabellentest: Beispiel

- In einem Warenwirtschaftssystem gelten folgende Geschäftsregeln:
  - Die Bestellmenge muss größer als Null sein
  - Teil-Lieferungen sind nicht erlaubt
  - Bei der Annahme einer Bestellung muss die Lagermenge entsprechend reduziert werden
  - Wird die Mindestmenge eines Lagerartikels unterschritten, muss eine Nachbestellung erfolgen

Textteil

Regelteil

Bestellmenge > 0	N	J	J	J
Bestellmenge > Art-Lagermenge	-	J	N	N
Art-Lagermenge - Bestellmenge ≥ Art-Mindestmenge	-	-	N	J
Melde "Bestellmenge ungültig"	X			
Melde "Menge nicht ausreichend"		X		
Reduziere Lagermenge			X	X
Schreibe Nachbestellung			X	

Bedingungsanzeiger:  
N = nicht erfüllt  
J = erfüllt  
- = ohne Bedeutung  
# = nicht definiert

Aktionsanzeiger:  
X = ausführen  
„“ = nicht ausführen  
(auch „-“ )

# Analyse von Entscheidungstabellen

Bedingung 1	N	N
Bedingung 2	-	J
Bedingung 3	-	-
Aktion 1	x	x
Aktion 2	x	
Aktion 3		x



- ET *vollständig*, wenn bei n Bedingungen alle  $2^n$  Kombinationen enthalten sind
  - Spalten im oberen Teil betrachten
- ET *redundanzfrei*, wenn speziellere Bedingungen zu anderen Aktionen führen
  - Spalten im oberen und unteren Teil betrachten
- ET *widerspruchsfrei*, wenn gleiche Bedingungen zu gleichen Aktionen führen
  - Spalten im oberen und unteren Teil betrachten

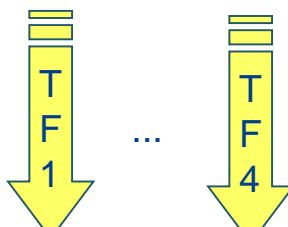
Bestellmenge > 0	N	N	N	N	J	J	J	J
Bestellmenge > Art-Lagermenge	N	N	J	J	N	N	J	J
Art-Lagermenge - Bestellmenge ≥ Art-Mindestmenge	N	J	N	J	N	J	N	J
Melde "Bestellmenge ungültig"	X	X	X	X				
Melde "Menge nicht ausreichend"							X	X
Reduziere Lagermenge					X	X		
Schreibe Nachbestellung					X			

Bestellmenge > 0	N	J	J	J
Bestellmenge > Art-Lagermenge	-	J	N	N
Art-Lagermenge - Bestellmenge ≥ Art-Mindestmenge	-	-	N	J
Melde "Bestellmenge ungültig"	X			
Melde "Menge nicht ausreichend"		X		
Reduziere Lagermenge			X	X
Schreibe Nachbestellung			X	

# Entscheidungstabellentest: Testfälle und -daten

- Jede Spalte (Regel) entspricht einem Testfall
  - Voraussetzungen pro Tabelle gleich
  - Bedingungen beziehen sich auf Eingaben
  - Aktionen spiegeln vorausgesagtes (erwartetes) Ergebnis wider
- Überdeckungskriterien z.B.
  - Alle Bedingungen mindestens einmal N und J
  - Alle Aktionen mindestens einmal x
  - Alle Spalten (alle Bedingungskombinationen)
- Konkrete Testdaten aus Wertebereichen ableiten
  - Äquivalenzklassenbildung
  - Grenzwertanalyse
  - ...

Ein Testfall pro Regel



B1	N	J	J	J
B2	-	N	N	J
B3	-	N	J	J
A1	x			
A2		x		x
A3			x	x
A4			x	x

#J ≥ 1;  
#N ≥ 1

#x ≥ 1

# Bewertung des Entscheidungstabellentests

- Vorteile
  - Kombinationen von Bedingungen, die sonst evtl. ungetestet blieben
  - Entscheidungstabellen-Technik allgemein geeignet, wenn Abläufe der Software von mehreren logischen Entscheidungen abhängen
  - Logische Zusammenhänge systematisch formulierbar
  - Entscheidungstabellen einfach auf Redundanz, Widerspruchsfreiheit und Vollständigkeit prüfbar
  - Zwingen nicht zur Strukturierung eines Ablaufs
  - Anwendbar auch bei einfacheren zustandsabhängigen Problemen
- Nachteile
  - Unübersichtlich bei zu vielen Bedingungen
  - Zusammenhänge zwischen einzelnen Bedingungen nur implizit ausdrückbar (vgl. Ursache-Wirkungs-Graph-Analyse)

**4.1**  
**Dynamischer**  
**Test**  
**Black-Box**

Dynamischer Test – Grundlagen

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

Zustandsbasierter Test

Entscheidungstabellentest

Anwendungsfallbasierter Test

Weitere Black-Box-Testverfahren

# Anwendungsfallbasierter Test (1 von 3)

**Anwendungsfall** (Use Case) Eine Folge von Vorgängen in einem Dialog zwischen Akteur und einer Komponente oder einem System, die zu einem konkreten Ergebnis führen. Ein Akteur kann dabei ein Benutzer sein, oder irgendetwas, was Informationen mit dem System austauschen kann. (ISTQB Glossar V3.2)

Anwendungsfälle beschreiben die Prozessabläufe auf Grundlage der Verwendung

Sie umfassen:

- Vorbedingungen und Nachbedingungen
- Üblicherweise ein Hauptszenario (das wahrscheinlichste Szenario)
- Ggf. alternative Abläufe und Ausnahmefälle (Varianten)

Anwendungsfallbasierte Tests gut geeignet, um Fehler im Praxiseinsatz aufzudecken



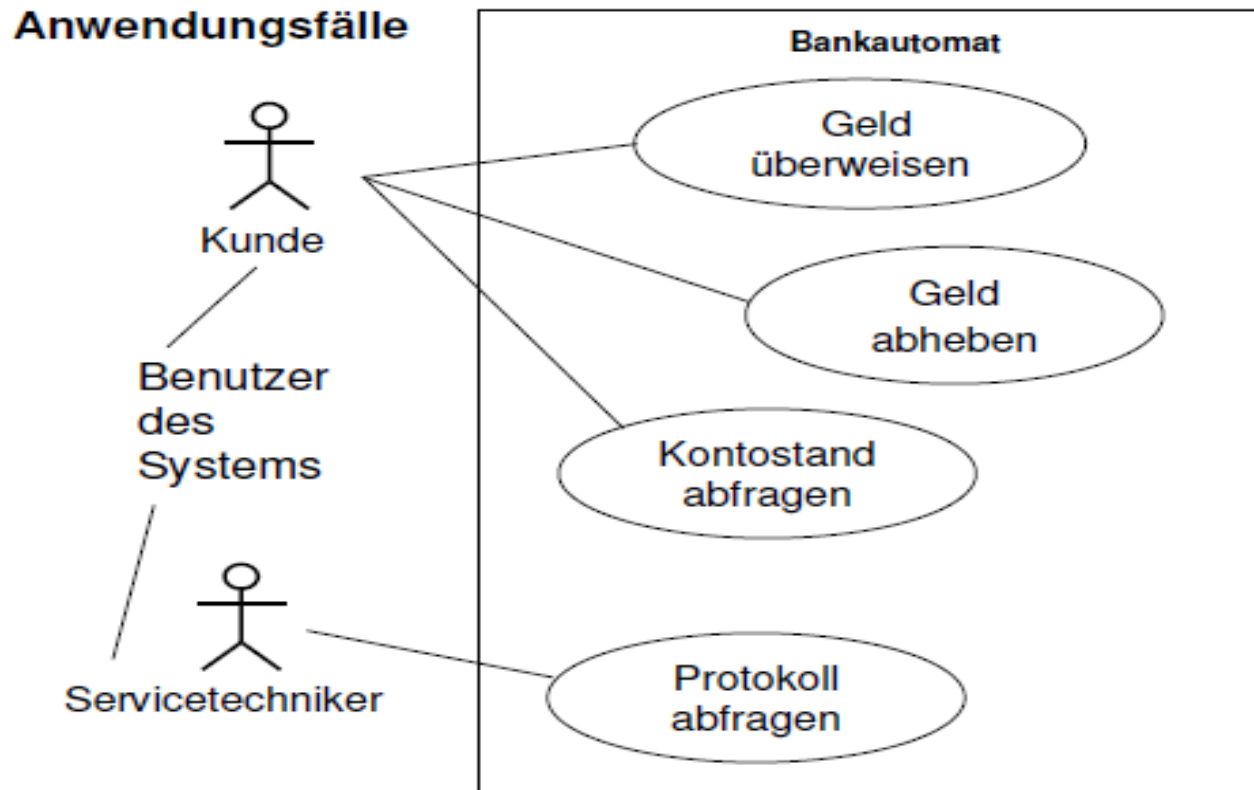
# Anwendungsfallbasierter Test (2 von 3)

## Anwendungsfallbasierter Test

Ein Black-Box-Testverfahren, bei dem Testfälle im Hinblick auf die Ausführung verschiedener Verhalten eines Anwendungsfalls entworfen werden. [UML 2.5.1] (ISTQB Glossar V3.2)

- Konkrete Abläufe von Anwendungsfällen (Szenarien) sind für den Entwurf von Abnahmetests mit Kunden-/Anwenderbeteiligung sehr hilfreich.
  - Betrachten Zusammenwirken und die gegenseitige Beeinflussung unterschiedlicher Komponenten
  - Dadurch können Fehler im Umfeld der Integration aufdeckt werden, die durch den Test der einzelnen Komponenten nicht gefunden werden

# Anwendungsfallbasierter Test (3 von 3) - Beispiel -



## **Anwendungsfall** Geld abheben **In Modell** Bankautomat

**Akteure** Kunde

**Vorbedingung** Kartenleser betriebsbereit **UND** Bedienpult gesperrt

### **Normaler Ablauf**

1. Der Kunde meldet sich am Bankautomaten an (include: Anmelden)
2. Der Kunde wählt als Transaktion Auszahlung
3. Der Kunde gibt den abzuhebenden Betrag ein
4. Der Bankautomat prüft den Betrag und meldet ihn an den Zentralrechner
5. Der Bankautomat aktualisiert die Karte und gibt sie aus
6. Der Kunde entnimmt die Karte
7. Der Bankautomat gibt das Geld aus
8. Der Kunde entnimmt das Geld

### **Alternativer Ablauf**

- 4.a Der Betrag ist zu hoch und muss neu eingegeben werden
- 4.b Der Betrag ist nicht in Scheinen auszahlbar und muss neu eingegeben werden

**Nachbedingung** Saldo des Kontos um Auszahlungsbetrag reduziert

**UND** Geldvorrat um den Auszahlungsbetrag reduziert

**UND** Kartenleser betriebsbereit **UND** Bedienpult gesperrt

### **Ausnahmeablauf**

- 6.a Die Karte wird nicht innerhalb von 60 Sekunden entnommen

**Nachbedingung** Karte eingezogen **UND** Kartenleser betriebsbereit **UND** Bedienpult gesperrt

### **Ausnahmeablauf**

- 1.-4. Der Kunde bricht den Vorgang ab
- 5.a Der Bankautomat aktualisiert die Karte und gibt sie aus
- 6.a Der Kunde entnimmt die Karte

**Nachbedingung** Kartenleser betriebsbereit **UND** Bedienpult gesperrt

**END** Auszahlung.

# Ablauforientierte Testfälle

- Testfälle so auswählen, dass die geforderte Überdeckung des Anwendungsfalls erzielt wird
  - Normaler Ablauf
  - Alternative Abläufe
  - Ausnahmeabläufe
  - Mögliche Wiederholungen innerhalb der Szenarien

# Kapitel 4

## 4.1 Dynamischer Test Black-Box

Dynamischer Test – Grundlagen

Idee der Black-Box-Testverfahren

Äquivalenzklassenbildung

Grenzwertanalyse

Zustandsbasierter Test

Entscheidungstabellentest

Anwendungsfallbasierter Test

Weitere Black-Box-Testverfahren

# Weitere Black-Box-Testverfahren

---

- Exkurs: Ursache-Wirkungs-Graph-Analyse
- Exkurs: Klassifikationsbaumverfahren
- Exkurs: Syntaxtest
- Exkurs: Zufallstest



# Exkurs: Ursache-Wirkungs-Graph-Analyse

- Graphische Beschreibung von (logischen) Wirkzusammenhängen
- Ursachen
  - Eingaben
  - Dateiinhalte / Datenbanken
  - Initiale Systemzustände
- Wirkungen
  - Ausgaben
  - Resultierende Systemzustände

Ursache A



Wirkung B

B = A

**Ursache-Wirkungs-Graph:** Eine graphische Darstellung der Eingaben und/oder Auslöser (Ursachen) und der zugeordneten Ausgaben (Wirkungen), die für den Entwurf von Testfällen verwendet werden können.

**Ursache-Wirkungs-Graph-Analyse:** Ein Black-Box-Testentwurfsverfahren, bei dem die Testfälle unter Nutzung des Ursache-Wirkungs-Graphen entworfen werden. [BS 7925/2]

# Ursache-Wirkungs-Graphen: Beispiel

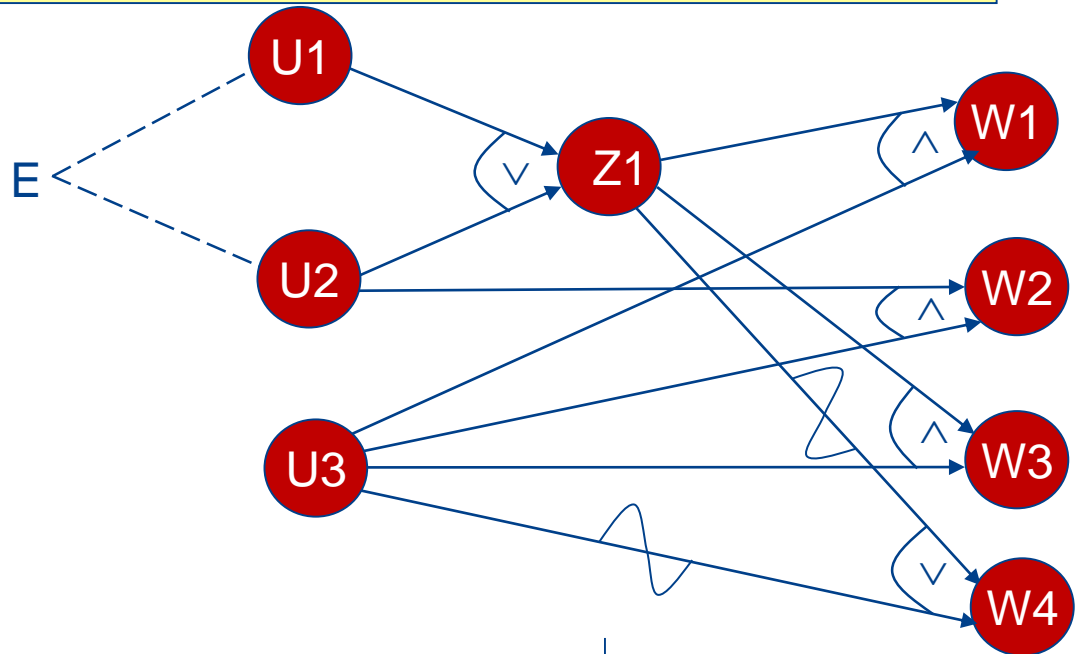
```
zaehle(gesamtzahl, vokalAnzahl: int)
```

- Liest solange Zeichen, bis ein Zeichen erkannt wird, das kein Großbuchstabe ist, oder `gesamtzahl` den größten `int`-Wert erreicht hat
- Ist gelesenes Zeichen großer Konsonant oder Vokal, wird `gesamtzahl` um 1 erhöht
- Ist der Großbuchstabe ein Vokal, wird `vokalAnzahl` um 1 erhöht
- Bei Beendigung werden `gesamtzahl` und `vokalAnzahl` zurückgegeben

U1: Großer Konsonant  
U2: Großer Vokal  
U3: `gesamtzahl < MaxCardinal`

W1: Gesamtzahl erhöhen  
W2: Vokalanzahl erhöhen  
W3: Zeichen lesen  
W4: Programmende

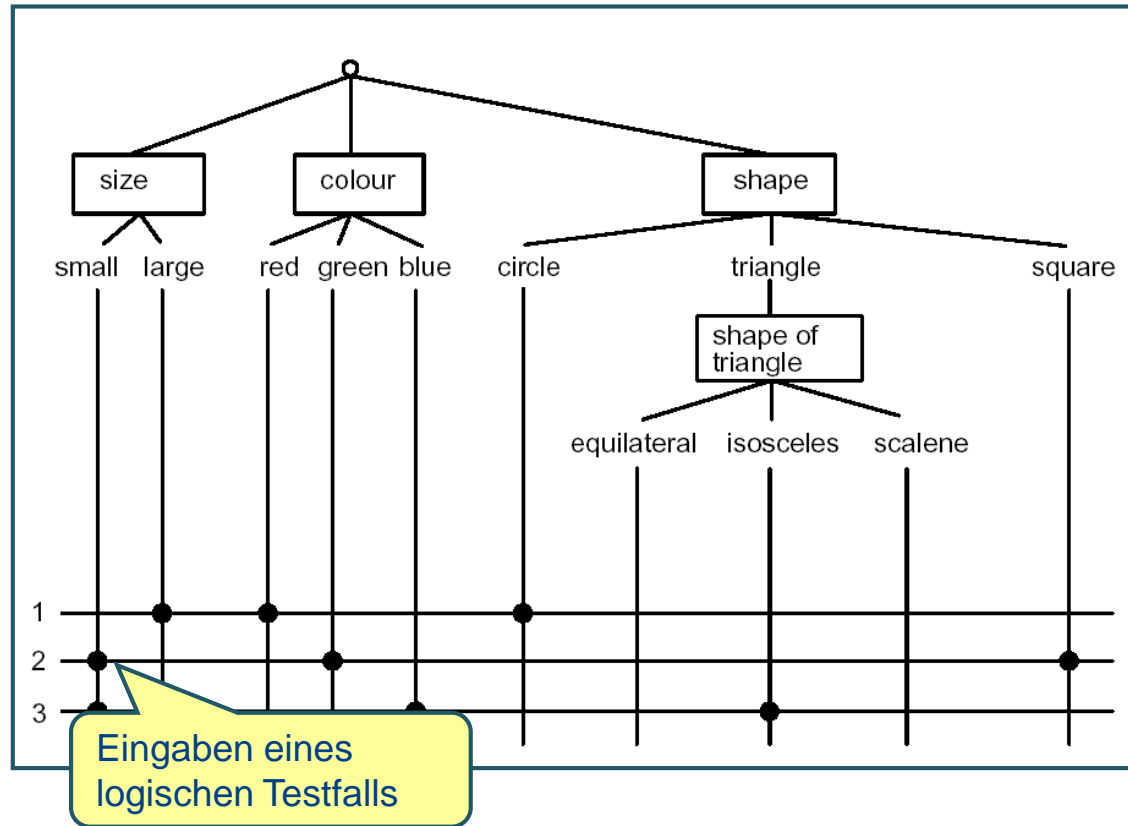
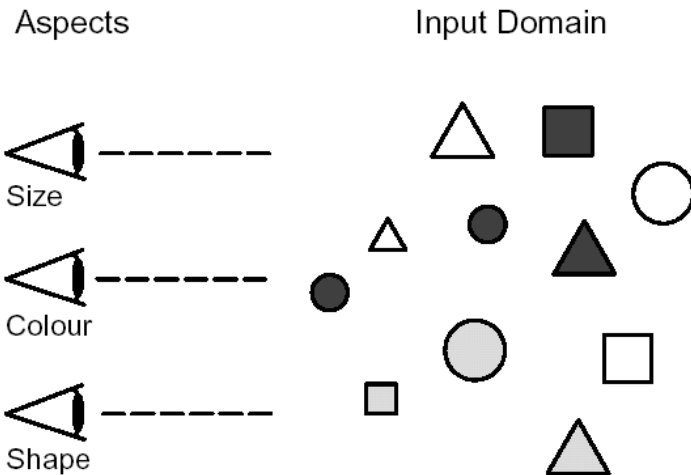
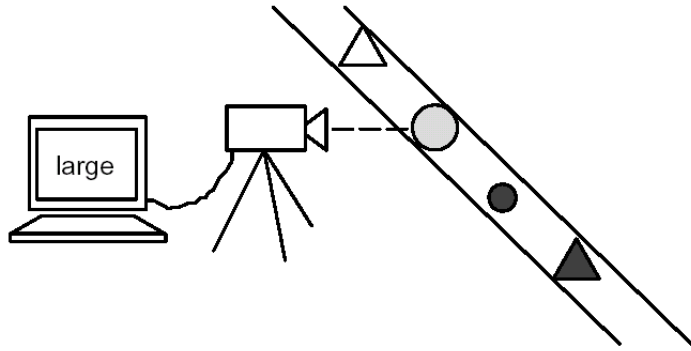
$\wedge$  logisches und  
 $\vee$  logisches oder  
~ logisches nicht





# Exkurs: Klassifikationsbäume

**Klassifikationsbaumverfahren:** Ein Black-Box-Testentwurfsverfahren, bei dem die durch einen Klassifikationsbaums dargestellten Testfälle so entworfen werden, dass Kombinationen der Repräsentanten von Eingabe- und/oder Ausgabebereichen (Äquivalenzklassen) ausgeführt werden. [Grochtmann] (ISTQB Glossar V3.2)



M. Grochtmann, J. Wegener, K. Grimm: Test Case Design Using Classification Trees and the Classification-Tree Editor CTE. Proc. Quality Week 1995



## Exkurs: Syntaxtest

### **Syntaxtest:**

Ein Black-Box-Testentwurfsverfahren, bei dem Testfälle auf Basis der Definition der Eingangsdaten erstellt werden. (ISTQB Glossar V3.2)

- Verfahren zur Ermittlung der Testfälle, das bei Vorliegen einer formalen Spezifikation der Syntax der Eingaben angewendet werden kann
- Die Regeln der syntaktischen Beschreibung werden genutzt, um Testfälle zu spezifizieren, welche sowohl die Einhaltung als auch die Verletzung der syntaktischen Regeln für die Eingaben berücksichtigen



## Beispiel: Syntaxtest (1 von 4)

- Float grammar in BNF

float = int "e" int.

int = ["+"|"-"] nat.

nat = {dig}.

dig = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".

- Tests auf gültige Syntax basierend auf der Bestimmung der Optionen
  - float hat keine Option
  - int hat drei Optionen: nat [opt\_1], "+" nat [opt\_2] und "-" nat [opt\_3]
  - nat hat zwei Optionen: eine einzelne Ziffer [opt\_4] und mehrere Ziffern [opt\_5]
  - dig hat zehn Optionen: eine für jede Ziffer [opt\_6 to opt\_15]
- Es sind 15 Optionen zu testen



## Beispiel: Syntaxtest (2 von 4)

Testfall	Eingabe	Option	Erwartetes Ergebnis
1	3e2	opt_1	Gültig
2	+2e+5	opt_2	Gültig
3	-6e-7	opt_3	Gültig
4	6e-2	opt_4	Gültig
5	1234567890e3	opt_5	Gültig
6	0e0	opt_6	Gültig
7	1e1	opt_7	Gültig
8	2e2	opt_8	Gültig
9	3e3	opt_9	Gültig
10	4e4	opt_10	Gültig
11	5e5	opt_11	Gültig
12	6e6	opt_12	Gültig
13	7e7	opt_13	Gültig
14	8e8	opt_14	Gültig
15	9e9	opt_15	Gültig

Keine minimale Testfallmenge ... es können Tests gewählt werden, die mehrere Optionen abdecken



## Beispiel: Syntaxtest (3 von 4)

- Test auf ungültige Syntax bspw. durch Mutation

- m1: Ungültiger Wert für ein Element
- m2: Ersetze ein Element durch ein anderes
- m3: Weglassen von Elementen
- m4: Hinzufügen von Elementen

- Nummerierung der Elemente zur Anwendung der Mutationen:

float = int "e" int.

int = ["+"|" -"] nat.

nat = {dig}.

dig = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".

*el\_1* = *el\_2 el\_3 el\_4*.

*el\_5* = *el\_6 el\_7*.

*el\_8* = *el\_9*.

*el\_10* = *el\_11*.



## Beispiel: Syntaxtest (4 von 4)

Testfall	Eingabe	Mutation	Element	Erwartetes Ergebnis
1	xe0	m1	x für el_2	Ungültig
2	0x0	m1	x für el_3	Ungültig
3	0ex	m1	x für el_4	Ungültig
4	x0e0	m1	x für el_6	Ungültig
5	+xe0	m1	x für el_7	Ungültig
6	ee0	m2	el_3 für el_2	Ungültig
7	+e0	m2	el_6 für el_2	Ungültig
8	000	m2	el_2 für el_3	Ungültig
9	0+0	m2	el_6 für el_3	Ungültig
10	0ee	m2	el_3 für el_4	Ungültig
11	0e+	m2	el_6 für el_4	Ungültig
12	e0e0	m2	el_3 für el_6	Ungültig
13	+ee0	m2	el_3 für el_7	Ungültig
14	++e0	m2	el_6 für el_7	Ungültig
15	e0	m3	el_2	Ungültig
16	00	m3	el_3	Ungültig
17	0e	m3	el_4	Ungültig
18	y0e0	m4	y in el_1	Ungültig
19	0ye0	m4	y in el_1	Ungültig
20	0ey0	m4	y in el_1	Ungültig
21	0e0y	m4	y in el_1	Ungültig
22	y+0e0	m4	y in el_5	Ungültig
23	+y0e0	m4	y in el_5	Ungültig
24	+0yeo	m4	y in el_5	Ungültig



## Exkurs: Zufallstest

### **Zufallstest:**

Ein Black-Box-Testentwurfsverfahren, bei dem Testfälle, unter Umständen unter Verwendung eines pseudozufälligen Generierungsalgorithmus, ausgewählt werden, um einem Nutzungsprofil in der Produktivumgebung zu entsprechen. (ISTQB Glossar V3.2)

- Der Generierungsalgorithmus wählt aus der Menge der möglichen Werte eines Eingabedatums zufällig Repräsentanten für die Testfälle aus.
- Pseudozufällig:
  - Basis der Auswahl der Repräsentanten:
    - Vermutete oder gegebene statistische Verteilung der Werte
  - Ziel:
    - möglichst realitätsnahe
      - Testfälle
      - Aussagen zur Zuverlässigkeit des Systems

# Bewertung der Black-Box-Testverfahren

- Grundlage: Anforderungen, Spezifikation von System, Komponenten und Schnittstellen
  - Fehlerhafte Anforderungen oder Spezifikationen werden nicht erkannt
  - Testobjekt verhält sich entsprechend der (evtl. fehlerhaften) Spezifikation
- Nicht geforderte Funktionalität wird nicht erkannt
  - Zusätzliche Funktionen sind nicht spezifiziert
  - Testfälle für zusätzliche Funktionen werden - wenn überhaupt - zufällig durchgeführt
  - Überdeckungskriterien nur auf Grundlage der Anforderungen bzw. der Spezifikation und nicht auf der Grundlage von nicht beschriebenen oder vermuteten Funktionen
  - nicht geforderte, evtl. schadhafte Funktionalität bleibt vermutlich unentdeckt
- Black-Box-Testverfahren auf Prüfung der Funktionalität des Testobjekts fokussiert
  - Black-Box-Tests für korrekte Funktion des Softwaresystems mit höchster Priorität





# Zusammenfassung



- Dynamische Tests führen das Testobjekt aus
- Spezieller Testrahmen oft notwendig
- Auswahl der Testfälle als (gute) Stichprobe
- Black-Box-Testverfahren benötigen zur Auswahl der Testfälle keine Kenntnis der Programmlogik
- Funktionale Tests leiten die Testfälle anhand der Spezifikation des Testobjekts ab
- Äquivalenzklassenbildung sollte in Kombination mit der Grenzwertanalyse zur Ermittlung der Testfälle eingesetzt werden
- Zustandsbasierte Tests sollten mit Übergangsbaum und Zustands-Konformanztest sowie erweitertem Übergangsbaum und Zustands-Robustheitstest eingesetzt werden
- Entscheidungstabellentest bei regelbasierten Anforderungen einsetzen
- Anwendungsfallbasierter Test für Szenarien der Systembenutzung



# Ergänzende Videos

- Black and White Box Testing Introduction  
<https://www.youtube.com/watch?v=jRwwb7iaRsU>
- Boundary Value Analysis  
<https://www.youtube.com/watch?v=5Ds5gpt0IXo>
- Boundary Value Analysis and Equivalence Partitioning  
<https://www.youtube.com/watch?v=P1Hv2sUPKeM>
- Grundlagen Testen  
<https://www.youtube.com/watch?v=yxcEhk0qsrc&t=45s>
- Whitebox testing and blackbox testing (German)  
<https://www.youtube.com/watch?v=nQQwBsK7YII>
- State Transition Testing – Tutorial  
[https://www.youtube.com/watch?v=Udjai\\_6b9Y](https://www.youtube.com/watch?v=Udjai_6b9Y)
- Testfälle mittels Äquivalenzklassen und Grenzwertanalyse  
<https://www.youtube.com/watch?v=GshMbff3mzw>
- Use Case Testing – Tutorial  
<https://www.youtube.com/watch?v=ijtvAvapsP0>
- What is Black Box Testing  
[https://www.youtube.com/watch?v=Uye2n\\_7G3rE](https://www.youtube.com/watch?v=Uye2n_7G3rE)
- What is Decision Table Testing  
[https://www.youtube.com/watch?v=TSbIDrx\\_KVY](https://www.youtube.com/watch?v=TSbIDrx_KVY)



# Folgende Fragen sollten Sie jetzt beantworten können

- Welches ist der wesentliche Unterschied zwischen dynamischen und statischen Tests?
- Welche grundlegenden Funktionen und Eigenschaften muss ein Testrahmen haben?
- Welches ist der wesentliche Unterschied zwischen Black-Box-Testverfahren und White-Box-Testverfahren zur Testfallermittlung?
- Worauf basieren funktionale Tests?
- Was versteht man unter der Äquivalenzklassenbildung?
- Wann ist die Grenzwertanalyse einsetzbar?
- Wann ist der zustandsbasierte Test einsetzbar?
- Wie ist eine Entscheidungstabelle aufgebaut?
- Wann ist der anwendungsfallbasierte Test einsetzbar?
- Welche weiteren Black-Box-Testverfahren gibt es noch?



# Muster-Prüfungsfragen

---

Testen Sie Ihr Wissen...



## Frage 1

25. Der Bonus eines Mitarbeiters soll berechnet werden. Der Bonus kann nicht negativ, aber 0 sein.

Der Bonus hängt von der Anstellungsdauer ab:

- Ein Mitarbeiter kann weniger als oder gleich 2 Jahre,
- mehr als 2 Jahre aber weniger als 5 Jahre,
- 5 bis inklusive 10 Jahre oder länger als 10 Jahre angestellt sein.

Wie viele Testfälle sind notwendig, wenn nur gültige Äquivalenzklassen für die Berechnung des Bonus verwendet werden [K3]?

a)	3	<input type="checkbox"/>
b)	5	<input type="checkbox"/>
c)	2	<input type="checkbox"/>
d)	4	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 1 - Lösung

25. Der Bonus eines Mitarbeiters soll berechnet werden. Der Bonus kann nicht negativ, aber 0 sein.

Der Bonus hängt von der Anstellungsdauer ab:

- Ein Mitarbeiter kann weniger als oder gleich 2 Jahre,
- mehr als 2 Jahre aber weniger als 5 Jahre,
- 5 bis inklusive 10 Jahre oder länger als 10 Jahre angestellt sein.

Wie viele Testfälle sind notwendig, wenn nur gültige Äquivalenzklassen für die Berechnung des Bonus verwendet werden [K3]?

a)	3	<input type="checkbox"/>
b)	5	<input type="checkbox"/>
c)	2	<input type="checkbox"/>
d)	4	<input checked="" type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 2

26. Ein Geschwindigkeitskontroll- und -berichtssystem hat folgende Eigenschaften:

Wenn Sie 50 km/h oder weniger fahren, passiert nichts.

Wenn Sie schneller als 50 km/h, aber 55 km/h oder weniger fahren, werden Sie verwart.

Wenn Sie schneller als 55 km/h, aber nicht mehr als 60 km/h fahren, müssen Sie eine Geldbuße bezahlen.

Wenn Sie schneller als 60 km/h fahren, wird Ihr Führerschein entzogen.

Die Geschwindigkeit in km/h liegt dem System als ganze Zahl vor.

Welcher wäre der wahrscheinlichste Satz von Werten (km/h), der durch die Grenzwertanalyse identifiziert wird, wobei nur die Grenzwerte auf den Grenzen der Äquivalenzklassen relevant sind? [K3]

a)	0, 49, 50, 54, 59, 60	<input type="checkbox"/>
b)	50, 55, 60	<input type="checkbox"/>
c)	49, 50, 54, 55, 60, 62	<input type="checkbox"/>
d)	50, 51, 55, 56, 60, 61	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 2 - Lösung

26. Ein Geschwindigkeitskontroll- und -berichtssystem hat folgende Eigenschaften:

Wenn Sie 50 km/h oder weniger fahren, passiert nichts.

Wenn Sie schneller als 50 km/h, aber 55 km/h oder weniger fahren, werden Sie verwahrt.

Wenn Sie schneller als 55 km/h, aber nicht mehr als 60 km/h fahren, müssen Sie eine Geldbuße bezahlen.

Wenn Sie schneller als 60 km/h fahren, wird Ihr Führerschein entzogen.

Die Geschwindigkeit in km/h liegt dem System als ganze Zahl vor.

Welcher wäre der wahrscheinlichste Satz von Werten (km/h), der durch die Grenzwertanalyse identifiziert wird, wobei nur die Grenzwerte auf den Grenzen der Äquivalenzklassen relevant sind? [K3]

a)	0, 49, 50, 54, 59, 60	<input type="checkbox"/>
b)	50, 55, 60	<input type="checkbox"/>
c)	49, 50, 54, 55, 60, 62	<input type="checkbox"/>
d)	50, 51, 55, 56, 60, 61	<input checked="" type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019





## Frage 3

27. Den Beschäftigten einer Firma wird nur dann eine Jahresprämie ausbezahlt, wenn sie länger als ein Jahr im Unternehmen beschäftigt sind und ein Ziel erreichen, das vorher individuell vereinbart wurde.

Dieser Sachverhalt lässt sich in einer Entscheidungstabelle darstellen:

Testfall-ID		T1	T2	T3	T4
Bedingung1	Beschäftigung länger als ein Jahr?	JA	NEIN	NEIN	JA
Bedingung2	Ziel vereinbart?	NEIN	NEIN	JA	JA
Bedingung3	Ziel erreicht?	NEIN	NEIN	JA	JA
Aktion	Auszahlung der Jahresprämie?	NEIN	NEIN	NEIN	JA

**Welcher der folgenden Testfälle für eine in der Praxis vorkommende Situation fehlt in der oben aufgeführten Entscheidungstabelle? [K3]**

a)	Bedingung1 = JA, Bedingung2 = NEIN, Bedingung3 = JA, Aktion = NEIN	<input type="checkbox"/>
b)	Bedingung1 = JA, Bedingung2 = JA, Bedingung3 = NEIN, Aktion = JA	<input type="checkbox"/>
c)	Bedingung1 = NEIN, Bedingung2 = NEIN, Bedingung3 = JA, Aktion = NEIN	<input type="checkbox"/>
d)	Bedingung1 = NEIN, Bedingung2 = JA, Bedingung3 = NEIN, Aktion = NEIN	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 3 - Lösung

27. Den Beschäftigten einer Firma wird nur dann eine Jahresprämie ausbezahlt, wenn sie länger als ein Jahr im Unternehmen beschäftigt sind und ein Ziel erreichen, das vorher individuell vereinbart wurde.

Dieser Sachverhalt lässt sich in einer Entscheidungstabelle darstellen:

Testfall-ID		T1	T2	T3	T4
Bedingung1	Beschäftigung länger als ein Jahr?	JA	NEIN	NEIN	JA
Bedingung2	Ziel vereinbart?	NEIN	NEIN	JA	JA
Bedingung3	Ziel erreicht?	NEIN	NEIN	JA	JA
Aktion	Auszahlung der Jahresprämie?	NEIN	NEIN	NEIN	JA

**Welcher der folgenden Testfälle für eine in der Praxis vorkommende Situation fehlt in der oben aufgeführten Entscheidungstabelle? [K3]**

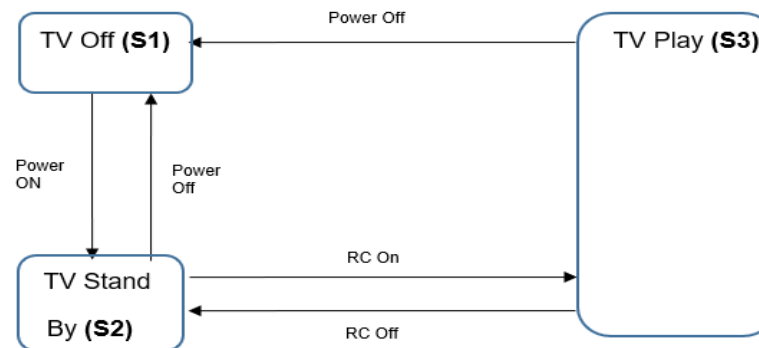
a)	Bedingung1 = JA, Bedingung2 = NEIN, Bedingung3 = JA, Aktion = NEIN	<input type="checkbox"/>
b)	Bedingung1 = JA, Bedingung2 = JA, Bedingung3 = NEIN, Aktion = JA	<input type="checkbox"/>
c)	Bedingung1 = NEIN, Bedingung2 = NEIN, Bedingung3 = JA, Aktion = NEIN	<input type="checkbox"/>
d)	Bedingung1 = NEIN, Bedingung2 = JA, Bedingung3 = NEIN, Aktion = NEIN	<input checked="" type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 4

28. Welche der folgenden Aussagen zum Zustandsdiagramm und der dargestellten Tabelle von Testfällen ist WAHR? [K3]



Testfall	1	2	3	4	5
Startzustand	S1	S2	S2	S3	S3
Eingabe	Power On	Power Off	RC On	RC Off	Power Off
Endzustand	S2	S1	S3	S2	S1

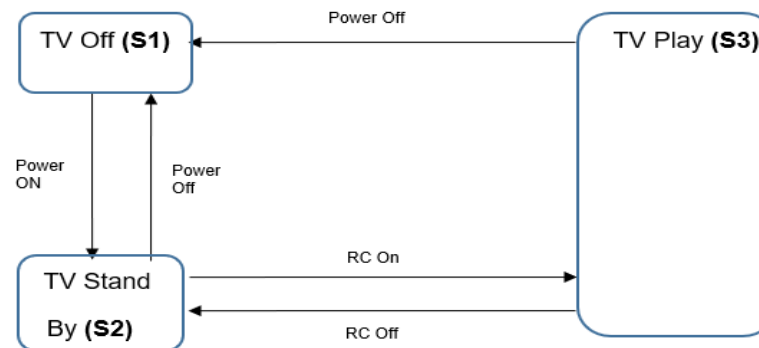
a)	Die Testfälle decken sowohl gültige als auch ungültige Übergänge des Zustandsdiagramms ab.	<input type="checkbox"/>
b)	Die Testfälle decken alle gültigen Übergänge des Zustandsdiagramms ab.	<input type="checkbox"/>
c)	Die Testfälle decken nur einige der möglichen Übergänge im Zustandsdiagramm ab.	<input type="checkbox"/>
d)	Die Testfälle decken Paare von Übergängen im Zustandsdiagramm ab.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 4 - Lösung

28. Welche der folgenden Aussagen zum Zustandsdiagramm und der dargestellten Tabelle von Testfällen ist WAHR? [K3]



Testfall	1	2	3	4	5
Startzustand	S1	S2	S2	S3	S3
Eingabe	Power On	Power Off	RC On	RC Off	Power Off
Endzustand	S2	S1	S3	S2	S1

a)	Die Testfälle decken sowohl gültige als auch ungültige Übergänge des Zustandsdiagramms ab.	<input type="checkbox"/>
b)	Die Testfälle decken alle gültigen Übergänge des Zustandsdiagramms ab.	<input checked="" type="checkbox"/>
c)	Die Testfälle decken nur einige der möglichen Übergänge im Zustandsdiagramm ab.	<input type="checkbox"/>
d)	Die Testfälle decken Paare von Übergängen im Zustandsdiagramm ab.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 5

**29. Eine Video-Anwendung hat folgende Anforderungen:**

**Die Anwendung soll die Wiedergabe eines Videos auf den folgenden Bildschirmauflösungen ermöglichen:**

- 1. 640x480.**
- 2. 1280x720.**
- 3. 1600x1200.**
- 4. 1920x1080.**

**Welcher der folgenden Testfall-Listen ist das Ergebnis der Anwendung der Äquivalenzklassenbildung? [K3]**

a)	Überprüfen, ob die Anwendung ein Video auf einem Display der Auflösung 1920x1080 abspielen kann (1 Testfall).	<input type="checkbox"/>
b)	Überprüfen, ob die Anwendung ein Video auf einem Display der Auflösung 640x480 und 1920x1080 abspielen kann (2 Testfälle).	<input type="checkbox"/>
c)	Überprüfen, ob die Anwendung ein Video auf jeder der geforderten Displayauflösung abspielen kann (4 Testfälle).	<input type="checkbox"/>
d)	Überprüfen, ob die Anwendung ein Video auf einer beliebigen der geforderten Displayauflösung abspielen kann (1 Testfall).	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



## Frage 5 - Lösung

**29. Eine Video-Anwendung hat folgende Anforderungen:**

**Die Anwendung soll die Wiedergabe eines Videos auf den folgenden Bildschirmauflösungen ermöglichen:**

- 1. 640x480.**
- 2. 1280x720.**
- 3. 1600x1200.**
- 4. 1920x1080.**

**Welcher der folgenden Testfall-Listen ist das Ergebnis der Anwendung der Äquivalenzklassenbildung? [K3]**

a)	Überprüfen, ob die Anwendung ein Video auf einem Display der Auflösung 1920x1080 abspielen kann (1 Testfall).	<input type="checkbox"/>
b)	Überprüfen, ob die Anwendung ein Video auf einem Display der Auflösung 640x480 und 1920x1080 abspielen kann (2 Testfälle).	<input type="checkbox"/>
c)	Überprüfen, ob die Anwendung ein Video auf jeder der geforderten Displayauflösung abspielen kann (4 Testfälle).	<input checked="" type="checkbox"/>
d)	Überprüfen, ob die Anwendung ein Video auf einer beliebigen der geforderten Displayauflösung abspielen kann (1 Testfall).	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019