

# Testen

Prof. Sven Apel

Universität des Saarlandes



# Teil III

Testtaktiken: Funktional

# Testtaktiken



Funktional  
"black box"

Tests basieren auf *Spezifikation*

Test deckt soviel *spezifiziertes*  
Verhalten wie möglich ab



Strukturell  
"white box"

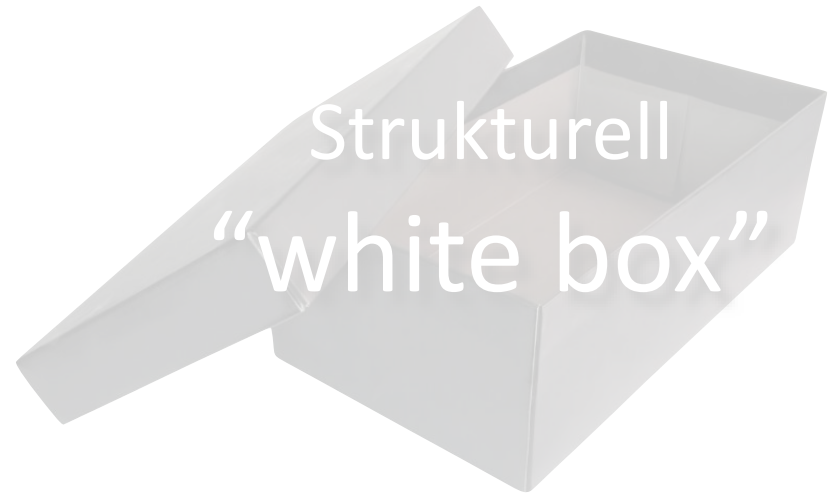
Tests basieren auf *Code*

Test deckt soviel *implementiertes*  
Verhalten wie möglich ab

# Testtaktiken



Funktional  
“black box”



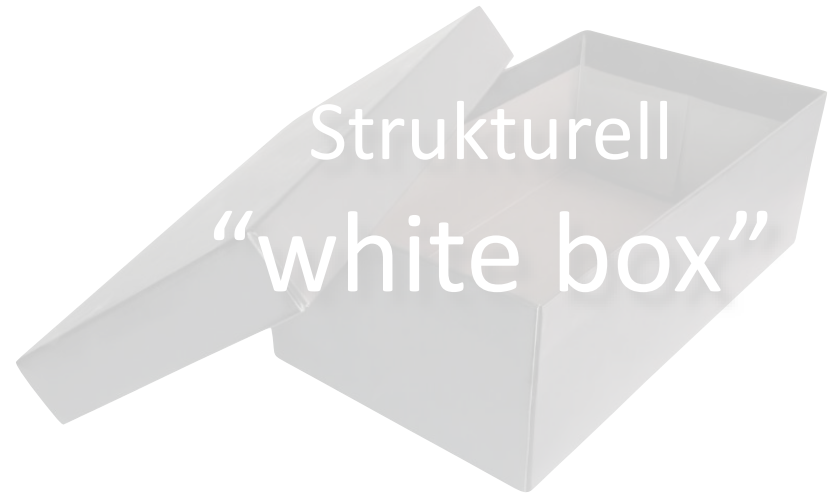
Strukturell  
“white box”

Code wird nicht benötigt

Frühes funktionales Testen hat Vorteile

zeigt Probleme der Spezifikation auf • prüft Testbarkeit • reichert  
Spezifikation mit weiteren Details an • Kostenreduktion!

# Testtaktiken



Am besten für Probleme *fehlender Logik*

Häufiges Problem: Details wurden einfach nicht implementiert

Funktionales Testen erkennt fehlenden Code

Anwendbar auf allen Testebenen

Unit-Tests • Integrationstests • Systemtests • Regressionstests

# Zufallstesten

Einfachste Art des funktionalen Testens

Wähle mögliche Eingaben gleichmäßig aus

Vermeidet *Voreingenommenheit* des Testers

Ein echtes Problem: Der Tester kann die gleichen logischen Fehler und falsche Annahmen machen wie der Programmierer (besonders, wenn es sich um dieselbe Person handelt)

Alle Eingaben sind gleichermaßen wertvoll

PUNKTE  
0



**Winkel**



**Kraft**





PUNKTE  
0









$2^{32} = 4.294.967.296$   
verschiedene Werte

Winkel



Kraft

$2^{32} = 4.294.967.296$   
verschiedene Werte

$2^{32} \equiv 4.294.967.296$   
verschiedene Werte

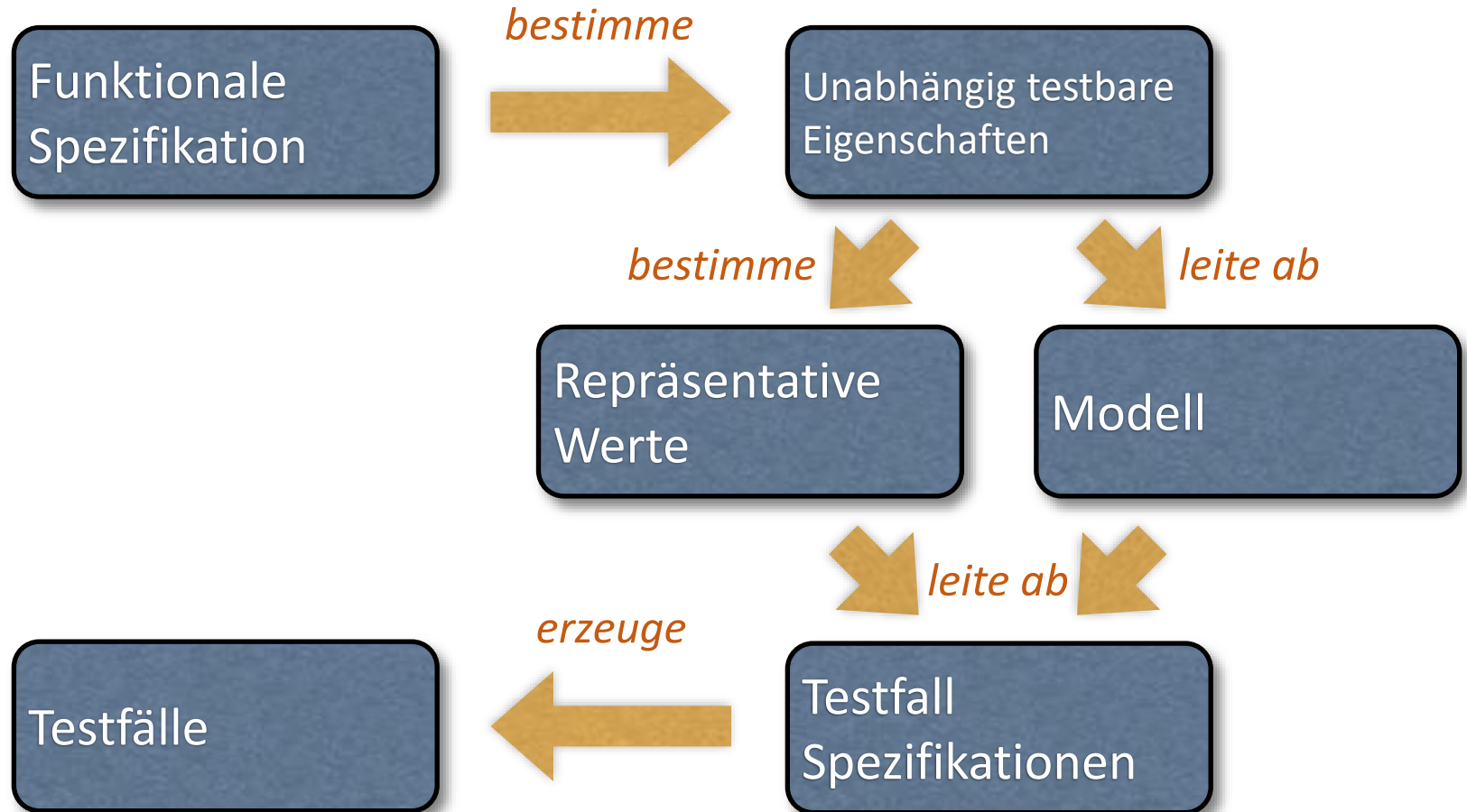


$2^{32} \equiv 4.294.967.296$   
verschiedene Werte

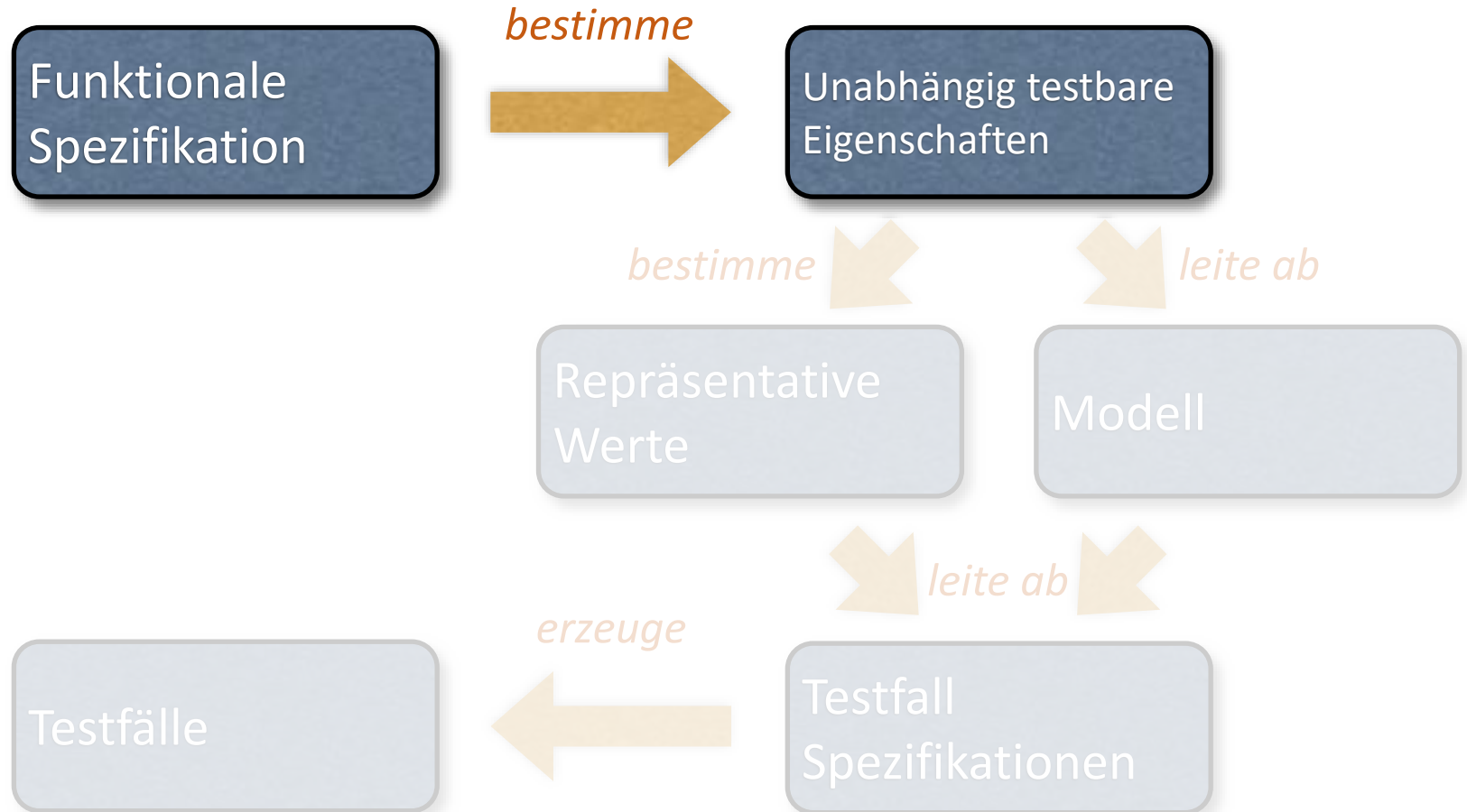


$2^{64} \equiv 18.446.744.073.709.551.616$   
verschiedene Läufe

# Systematisches Funktionales Testen



# Testbare Eigenschaften



# Testbare Eigenschaften



Teile System in *unabhängig testbare Eigenschaften* auf

Diese müssen keine Module oder Subsysteme der Software sein

Beim Systemtesten sind unabhängig testbare Eigenschaften durch Benutzeroberflächen oder APIs gegeben

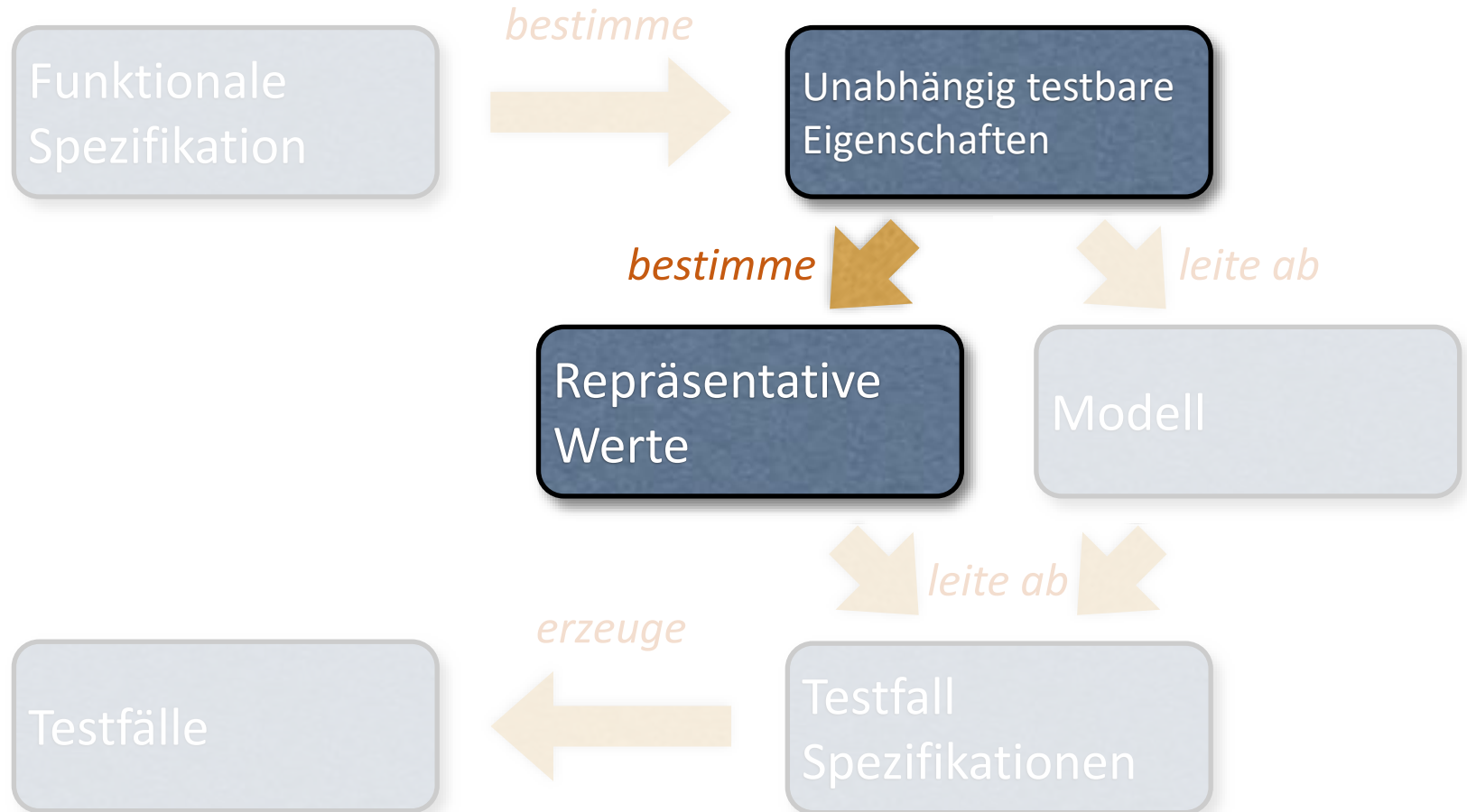


# Testbare Eigenschaften

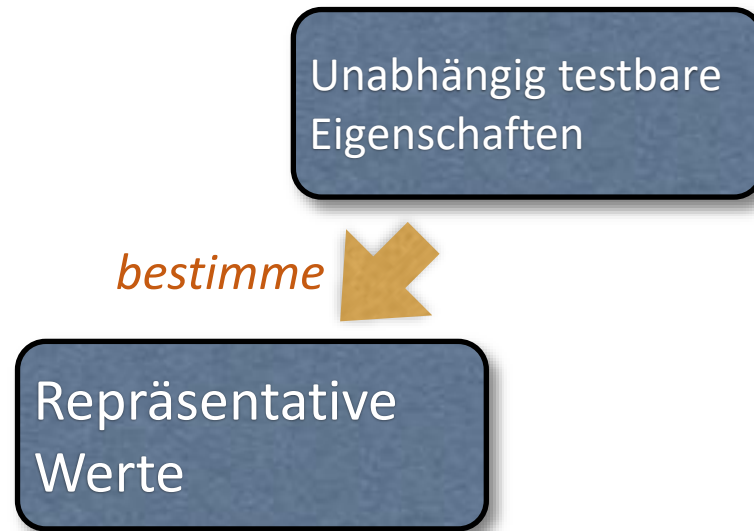
```
class Roots {  
  
    // Löse  $ax^2 + bx + c = 0$   
    public roots(double a, double b, double c) { ... }  
  
    // Ergebnis: Werte für x  
    double root_one, root_two;  
}
```

Was sind die unabhängig testbaren Eigenschaften?

# Repräsentative Werte



# Repräsentative Werte



Suche Eingaben, die *besonders wertvoll* sind

Typischerweise durch Wahl von *Repräsentanten*  
von *Äquivalenzklassen*

# Nadeln im Heuhaufen

Um Nadeln zu finden,  
müssen wir systematisch suchen

Wir müssen herausfinden,  
was *Nadeln besonders macht*



# Partitionstesten

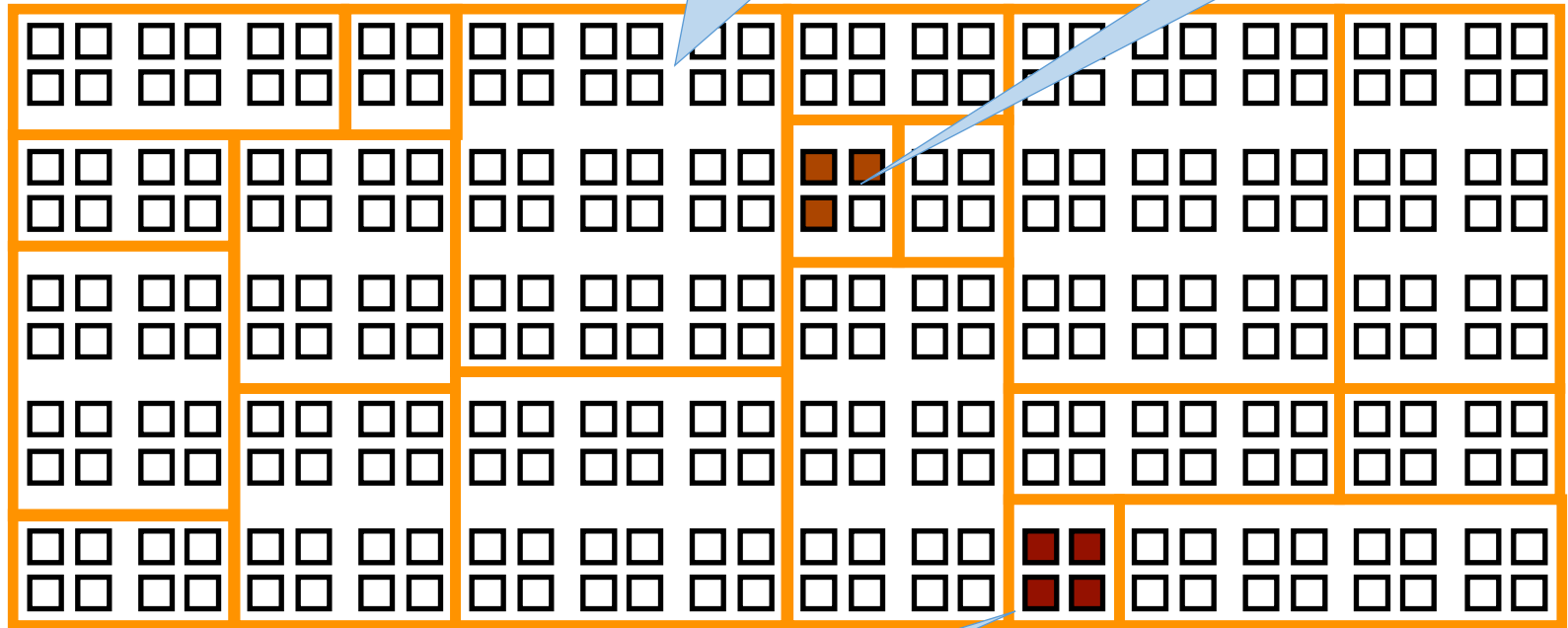
■ Fehlschlag (wertvoller Testfall)

□ Erfolg

Fehlschläge sind *selten*  
im Raum der möglichen  
Eingaben...

... aber *dicht* in  
einigen Teilen  
des Raums

Der Raum der möglichen Eingaben  
(der Heuhaufen)



Wenn wir systematisch jeden  
Teil des Raumes testen, testen  
wir auch die dichten Teile.

Funktionales Testen zieht *Linien* um  
*Regionen mit möglichen*  
*Fehlschlägen (= Äquivalenzklassen)*

# Äquivalenzklassen

Testen nimmt an, dass Programmierer *allgemeine Lösungen* implementieren

Diese Lösungen *induzieren Äquivalenzklassen* und machen so Testen erst möglich

Verletzungen dieser Annahme können durch Testen alleine nicht gefunden werden

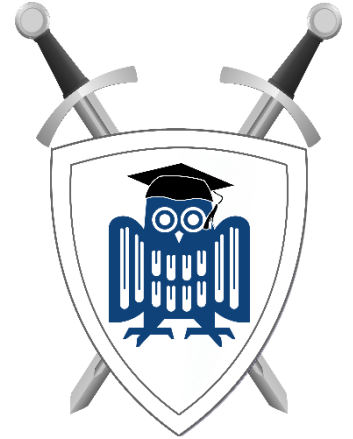
```
if (spieler.name().equals("Max")) score += 100;
```

# Annahmen an Äquivalenzklassen

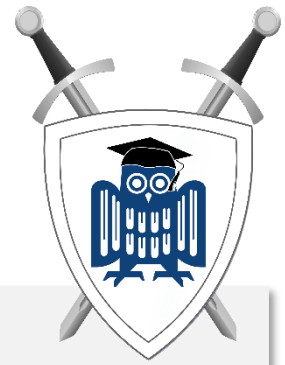
Beispiel:

Alle *Truhen* verhalten sich gleich  
→ es reicht, *eine* zu testen

Alle *Gegner* eines Typs verhalten sich gleich  
→ es reicht, *einen* zu testen



# Aus der SoPra-Spezifikation...



## 3.3. Spielfeld

Das Spielfeld des diesjährigen Spiels ist kein klassisch rechteckiges Feld, sondern besteht aus hexagonalen Feldern. Sie können also von einem Feld nach Westen, Osten, Nordwesten, Nordosten, Südwesten und Südosten gehen. Dies erlaubt die Gestaltung einer kreativeren Spielkarte.

## 3.6. Gegner

Neben dem bereits erwähnten Endgegner gilt es auf der Spielwelt auch noch andere Gegner, welche im Dienst des Professors stehen, zu besiegen. Folgend werden die verschiedenen Gegnertypen aufgelistet:

**Wächter:** Wächter sind jene Gegner, die auf der Insel, außerhalb der Dungeons, herumlaufen. Es gibt genau sieben dieser Wächter. Die Namen der Wächter sind Elias, Simone, Julius, Johanna, Janine, Nicole und Laura .

**Käfer:** ...

**Überläufer:** ...

...

# Welche Tests leiten sich aus daraus ab?

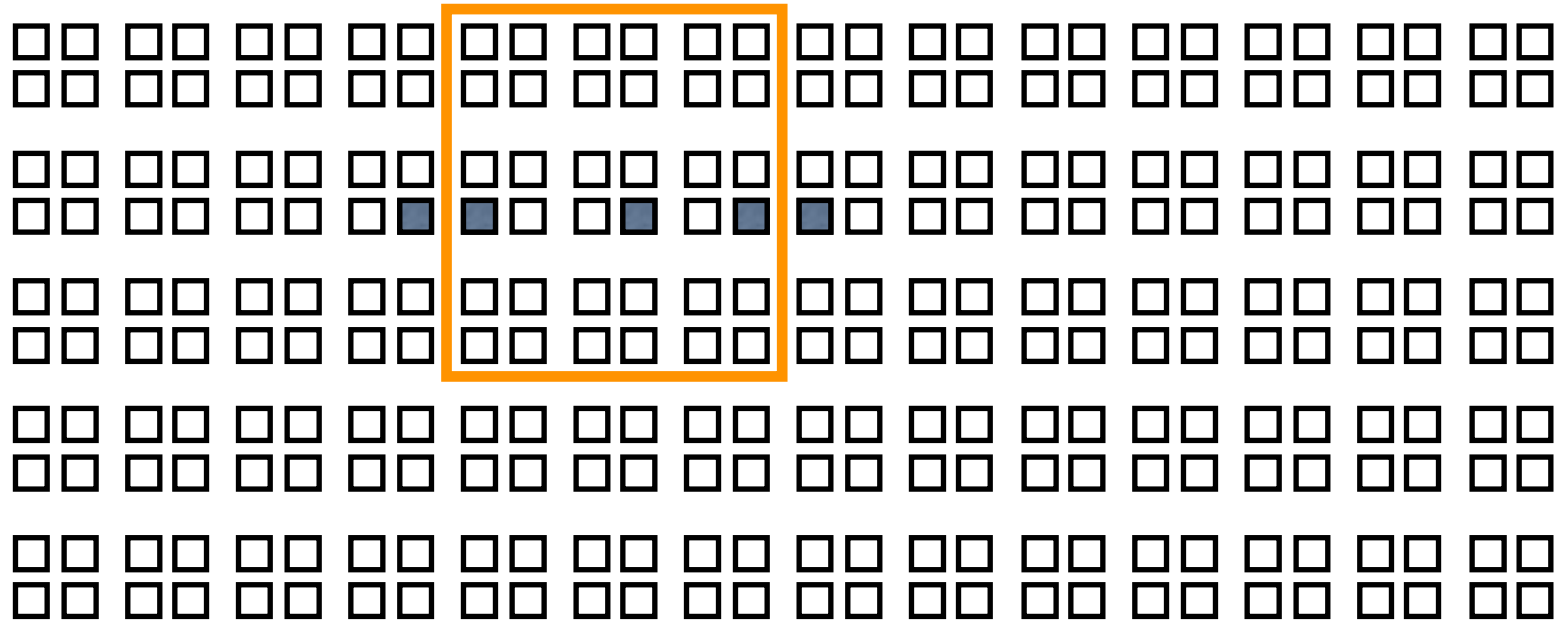


# Tests aus Bedingungen

Eingabebedingung	Äquivalenzklassen
Bereich	eine gültig, zwei ungültig (größer und kleiner)
gegebener Wert	eine gültig, zwei ungültig (größer und kleiner)
Element einer Menge	eine gültig, eine ungültig
Boolean	eine gültig, eine ungültig

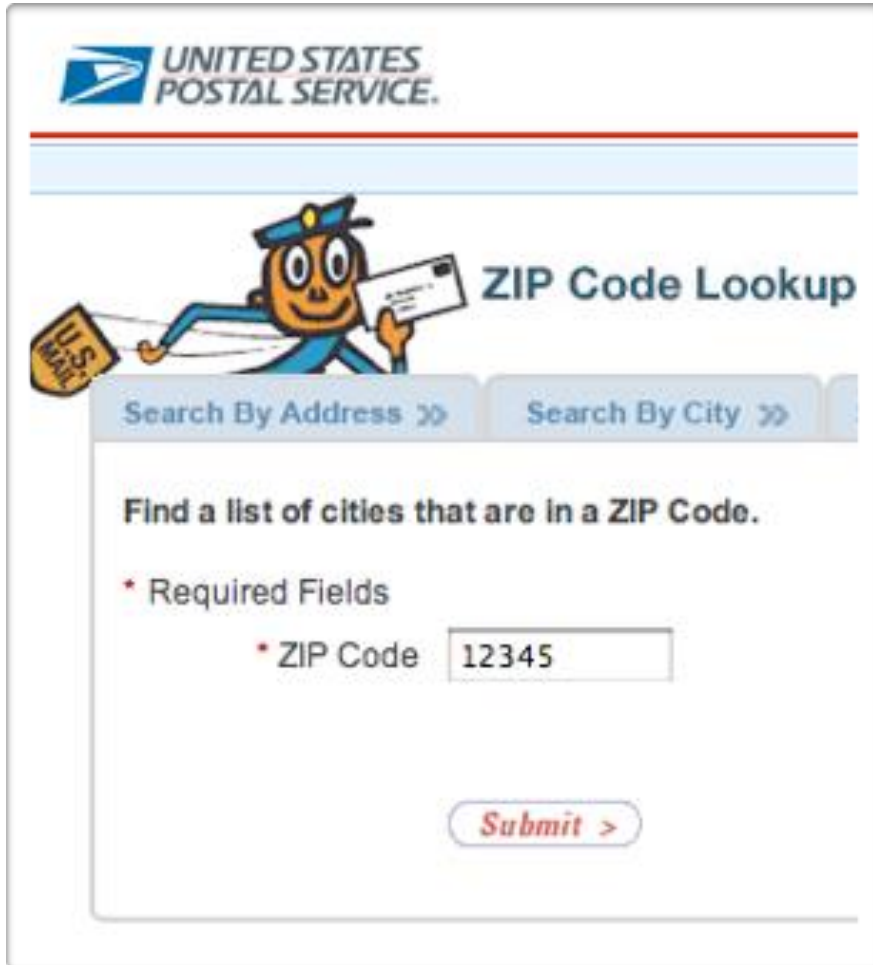
# Grenzanalyse

□ Möglicher Testfall



Test am *unteren Ende* (1x gültig, 1x ungültig),  
am *oberen Ende* (1x gültig, 1x ungültig),  
und *in der Mitte*

# Beispiel: Postleitzahl



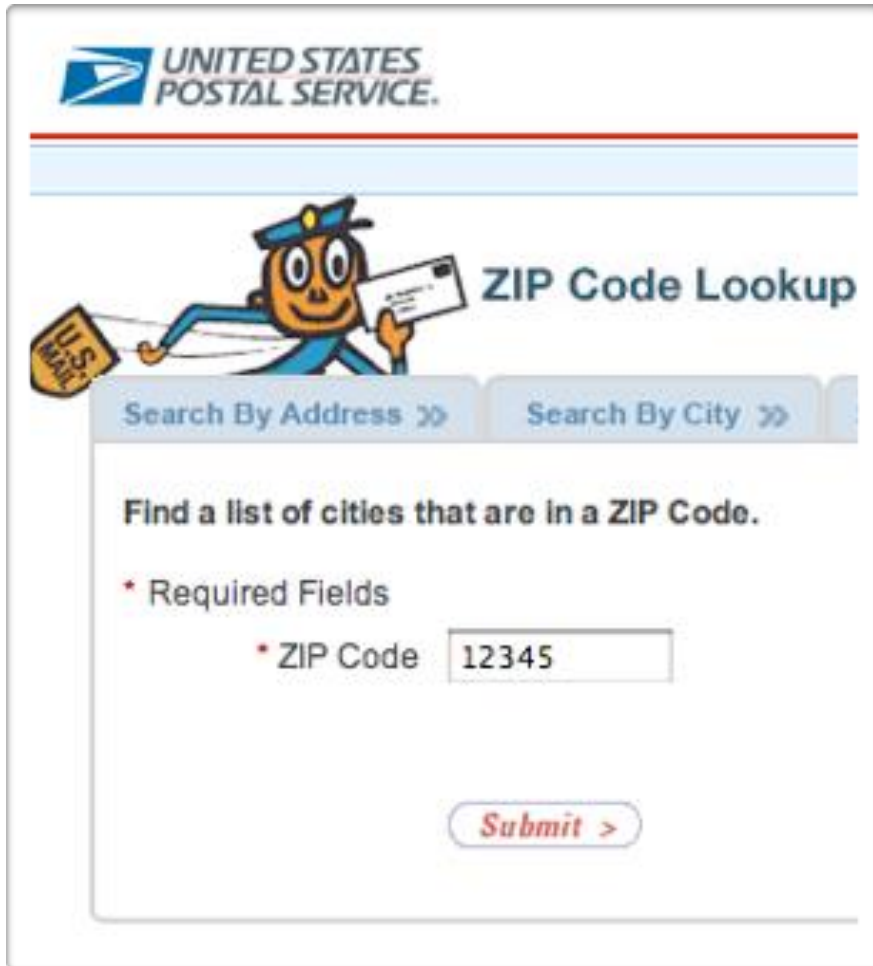
The image shows a screenshot of the United States Postal Service's ZIP Code Lookup page. At the top left is the USPS logo with the text "UNITED STATES POSTAL SERVICE.". Below the logo is a cartoon mailman holding a letter and a "U.S. Mail" sign. To the right of the mailman is the title "ZIP Code Lookup". Below the title are two buttons: "Search By Address >>" and "Search By City >>". Below these buttons is a text prompt: "Find a list of cities that are in a ZIP Code.". Underneath this prompt is a section titled "\* Required Fields". In this section, there is a label "\* ZIP Code" followed by a text input field containing the value "12345". At the bottom of the form is a red "Submit >" button.

Eingabe:  
5-stellige Postleitzahl

Ausgabe:  
Liste der Städte

Welche repräsentativen Werte nutzen wir zum Testen?

# Gültige Postleitzahlen



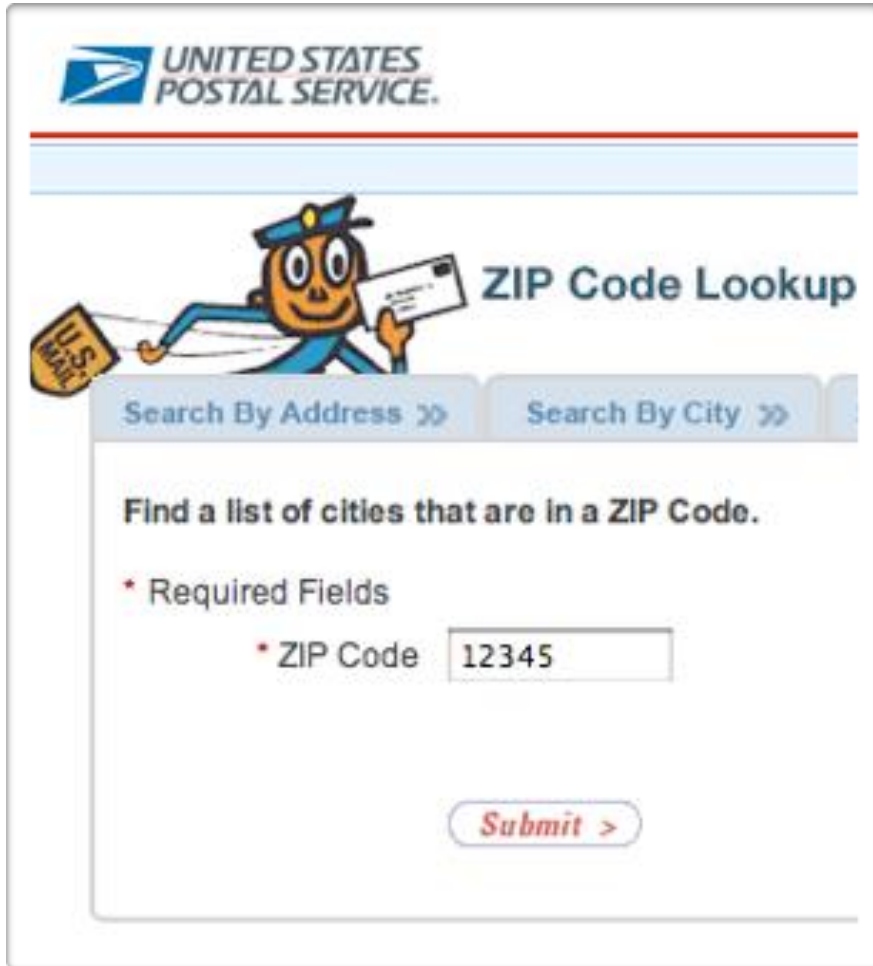
The image shows a screenshot of the United States Postal Service's ZIP Code Lookup web page. At the top left is the USPS logo with the text "UNITED STATES POSTAL SERVICE.". Below the logo is a cartoon mailman character holding a letter and a "U.S. Mail" sign. To the right of the mailman is the title "ZIP Code Lookup". Below the title are two buttons: "Search By Address >>" and "Search By City >>". Below these buttons is the instruction "Find a list of cities that are in a ZIP Code.". Underneath this is a section titled "\* Required Fields" with a label "\* ZIP Code" next to a text input field containing the value "12345". At the bottom of the form is a red "Submit >" button.

1. Mit 0 Städten als Ausgabe (0 ist Grenzwert)

2. Mit 1 Stadt als Ausgabe

3. Mit  $\geq 2$  Städten als Ausgabe

# Ungültige Postleitzahlen



UNITED STATES  
POSTAL SERVICE.

**ZIP Code Lookup**

Search By Address >> Search By City >>

Find a list of cities that are in a ZIP Code.

\* Required Fields

\* ZIP Code

**Submit >**

4. Leere Eingabe

5. 1–4 Zeichen (4 ist Grenzwert)

6. 6 Zeichen (6 ist Grenzwert)

7. sehr lange Eingabe

8. keine Ziffern

9. ungültige Zeichen

# “Besondere” Postleitzahlen

Wie wäre es mit einer Postleitzahl, die da lautet

```
12345'; DROP TABLE orders; SELECT * FROM zipcodes WHERE 'zip' = '
```

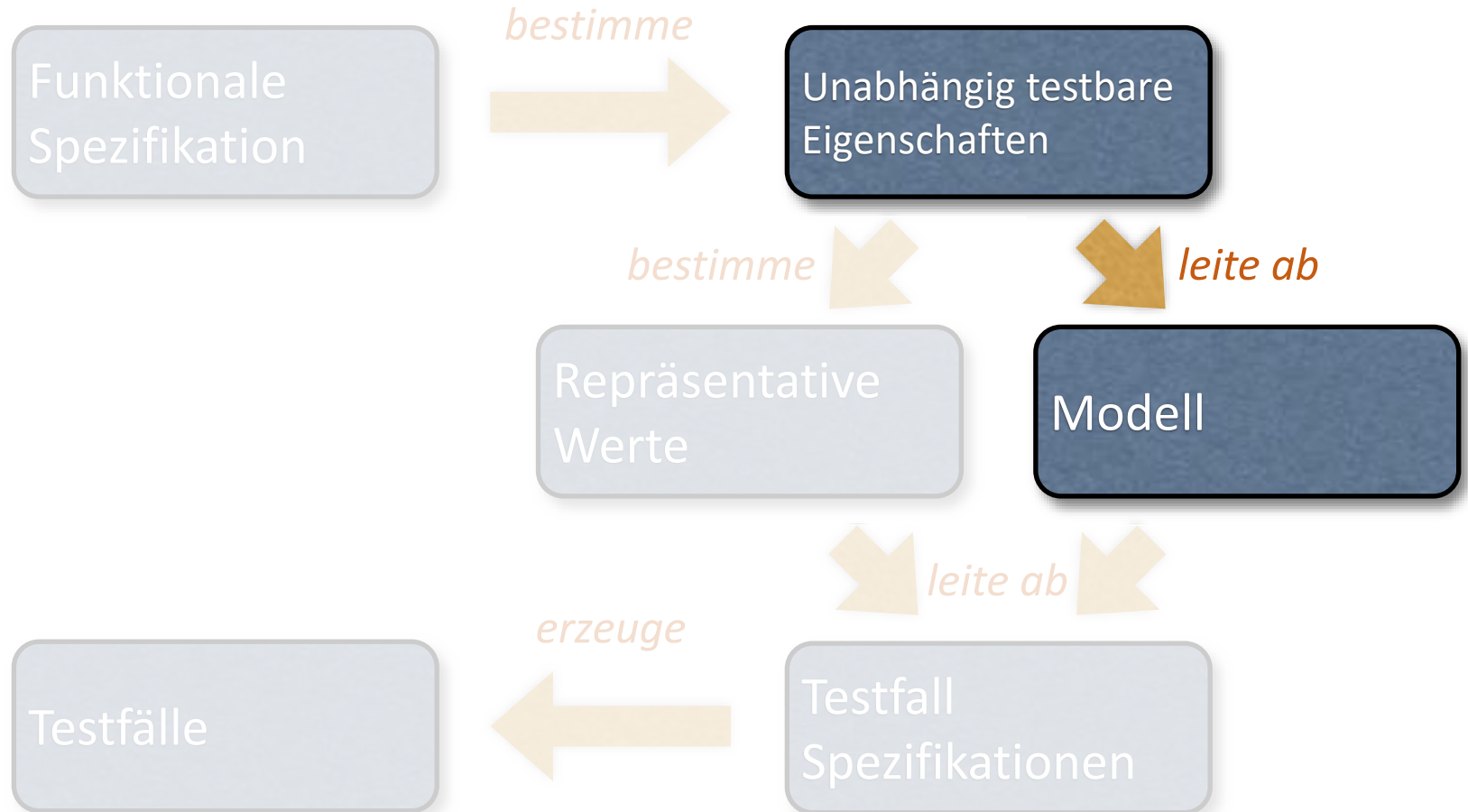
Oder eine Postleitzahl aus 65536 Zeichen...

Dies ist Sicherheitstesten!

# Gutjahrs Hypothese

Partitionstesten  
ist effektiver  
als Zufallstesten.

# Modellbasiertes Testen





# Modellbasiertes Testen

Ein *formales Modell* spezifiziert Software-Verhalten

Typische Arten von Modellen:

*Endliche Automaten* und  
*Entscheidungsstrukturen*

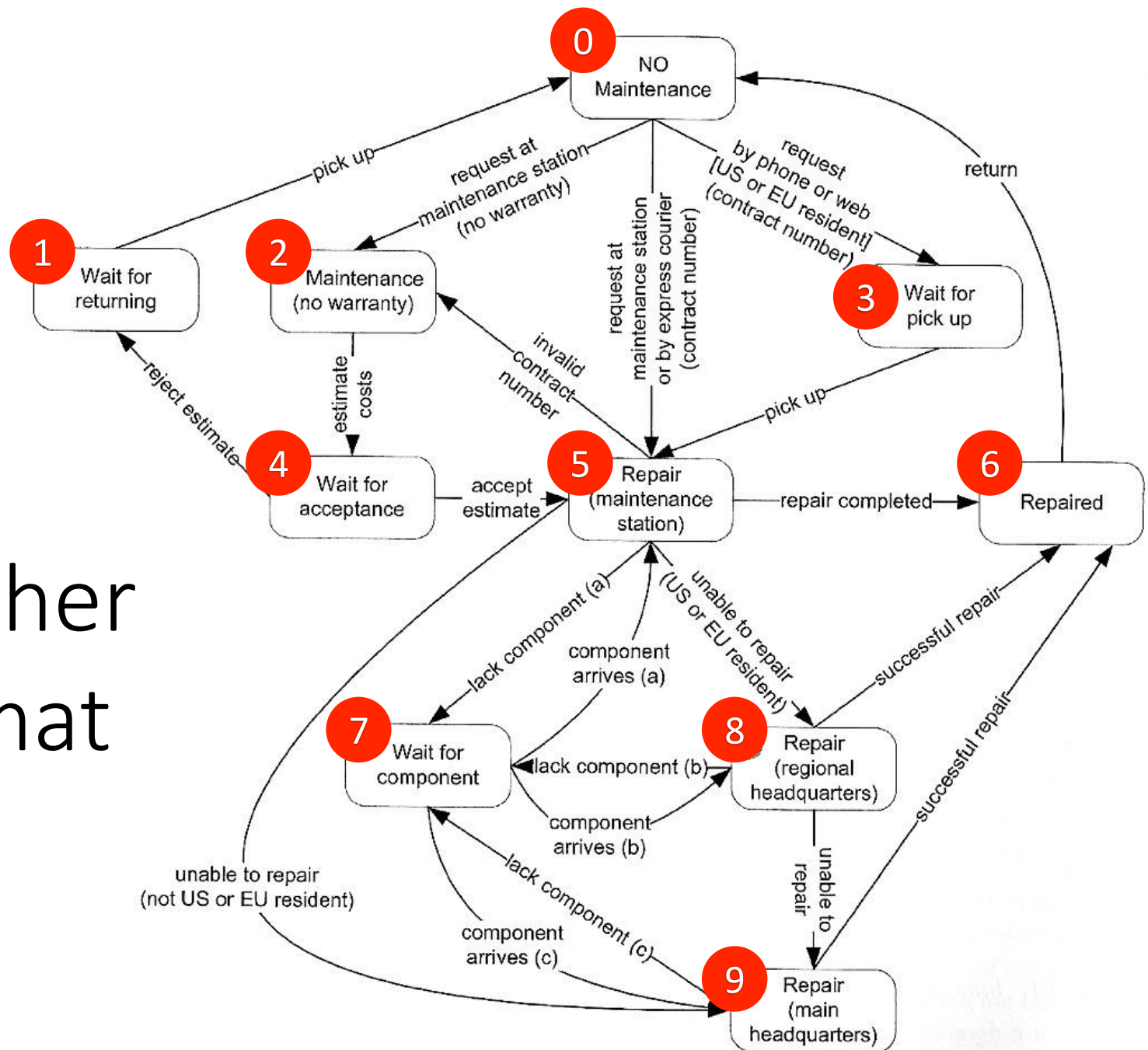
Unabhängig testbare  
Eigenschaften



*leite ab*

Modell

# Endlicher Automat



**Maintenance:** The *Maintenance* function records the history of items undergoing maintenance.

If the product is covered by warranty or maintenance contract, maintenance can be requested either by calling the maintenance toll free number, or through the Web site, or by bringing the item to a designated maintenance station.

If the maintenance is requested by phone or Web site and the customer is a US or EU resident, the item is picked up at the customer site, otherwise, the customer shall ship the item with an express courier.

If the maintenance contract number provided by the customer is not valid, the item follows the procedure for items not covered by warranty.

If the product is not covered by warranty or maintenance contract, maintenance can be requested only by bringing the item to a maintenance station. The maintenance station informs the customer of the estimated costs for repair. Maintenance starts only when the customer accepts the estimate. If the customer does not accept the estimate, the product is returned to the customer.

Small problems can be repaired directly at the maintenance station. If the maintenance station cannot solve the problem, the product is sent to the maintenance regional headquarters (if in US or EU) or to the maintenance main headquarters (otherwise).

If the maintenance regional headquarters cannot solve the problem, the product is sent to the maintenance main headquarters.

Maintenance is suspended if some components are not available.

Once repaired, the product is returned to the customer.

# Abdeckungskriterien

*Pfadabdeckung:* Tests decken jeden Pfad ab

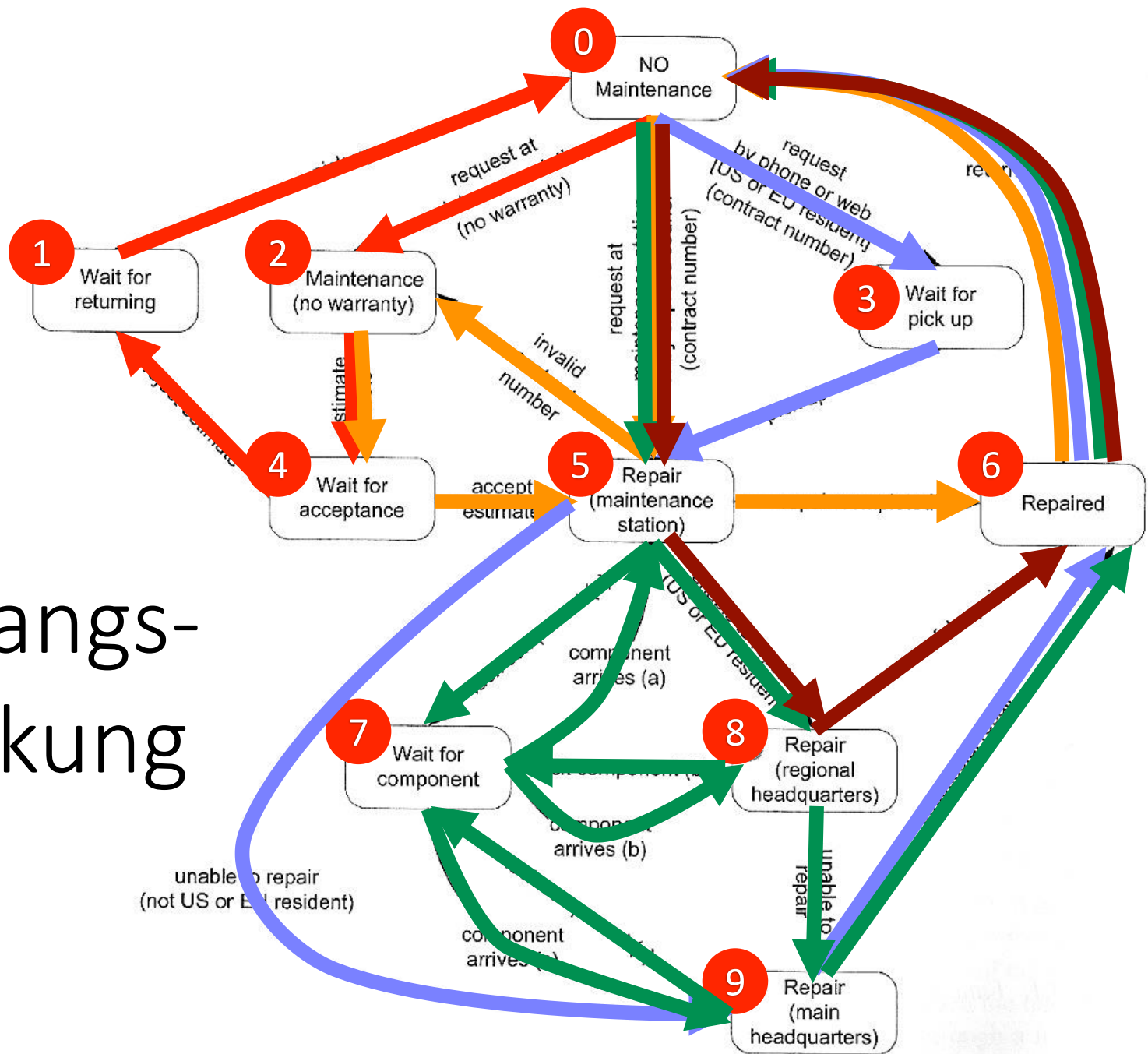
In der Praxis nicht möglich, da unendlich viele Pfade

*Zustandsabdeckung:* Jeder *Knoten* wird erreicht

Mindestkriterium zum Testen

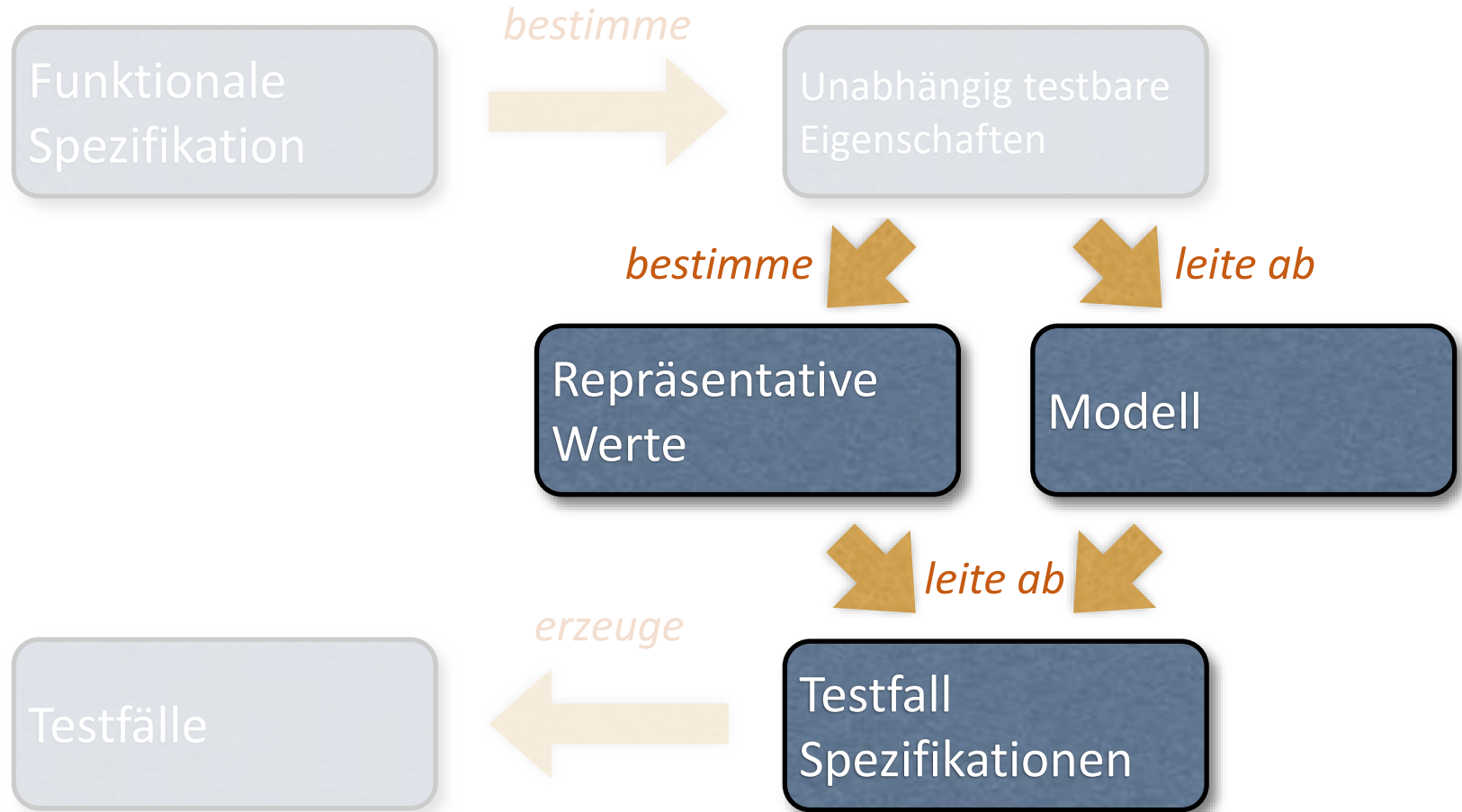
*Übergangsabdeckung:* Jede *Kante* wird erreicht

Sinnvolles und gewöhnlich erfüllbares Abdeckungskriterium



Übergangs-  
Abdeckung

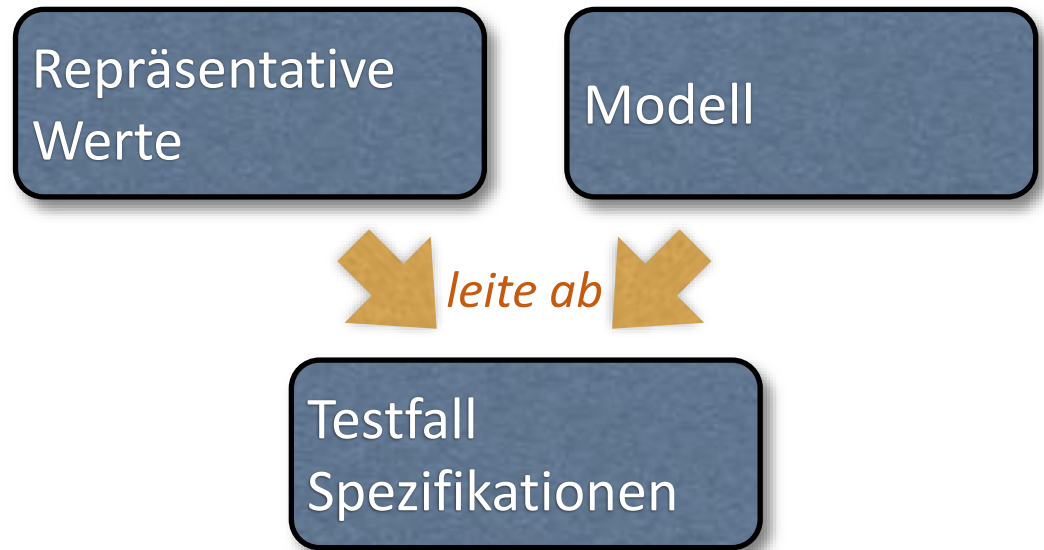
# Spezifikation von Testfällen



# Spezifikation von Testfällen

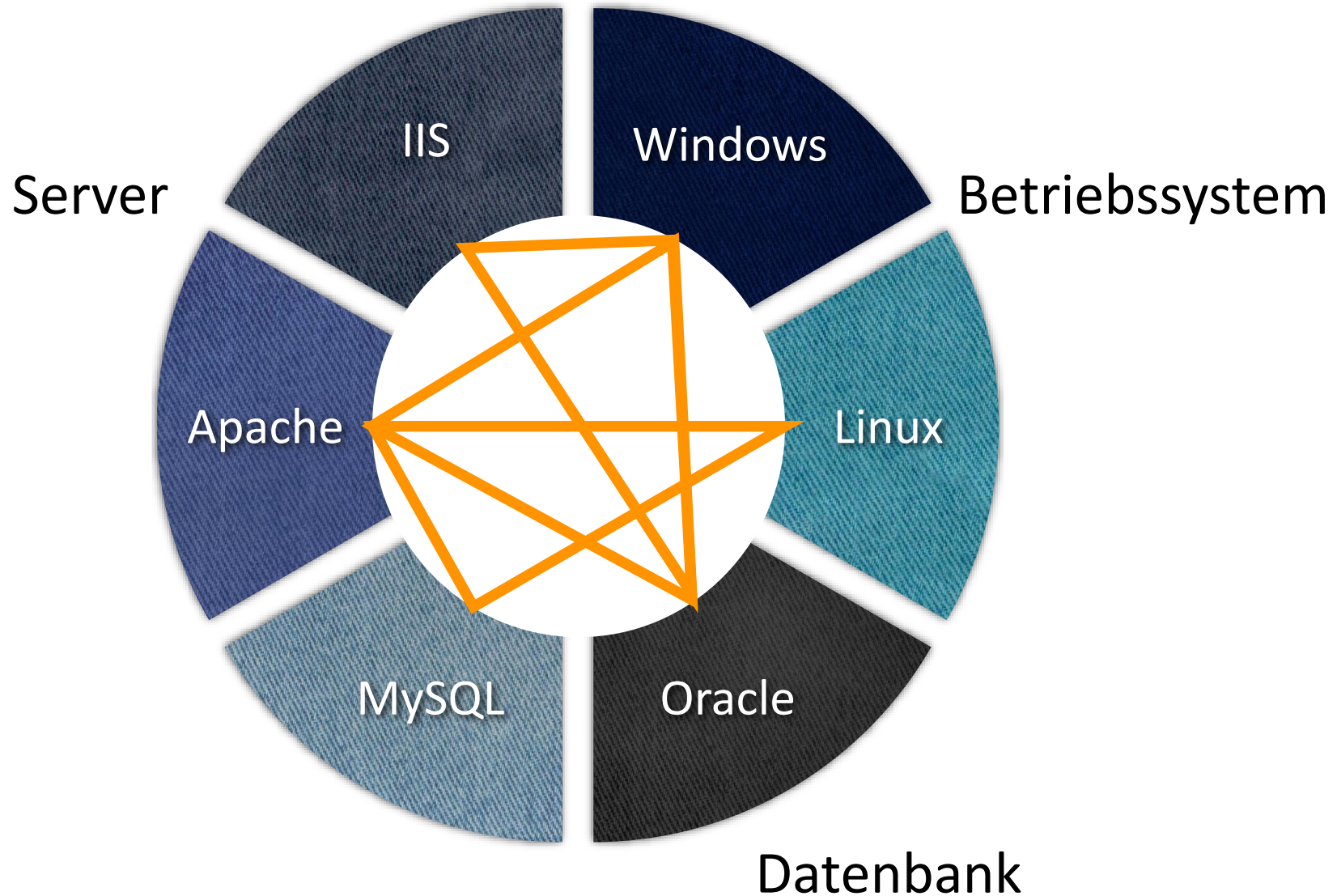
Eingabewerte in vorangegangenen Schritten aufgezählt und *kombinieren*!

Nutze Modelle  
*und* repräsentative  
Werte, um Testfälle  
abzuleiten





# Kombinatorisches Testen





# Kombinationstesten

*Ungültige* Kombinationen ausschließen

IIS etwa läuft nur auf Windows

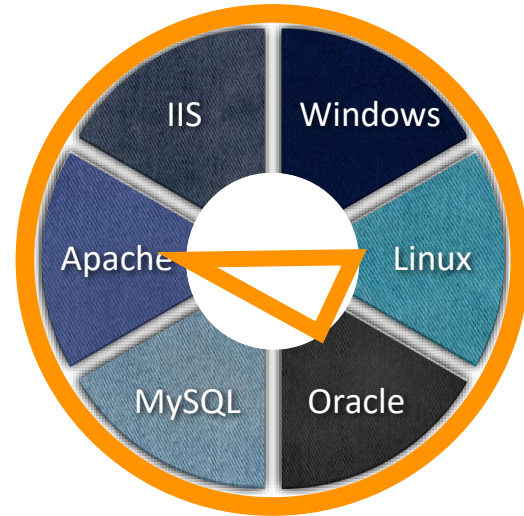
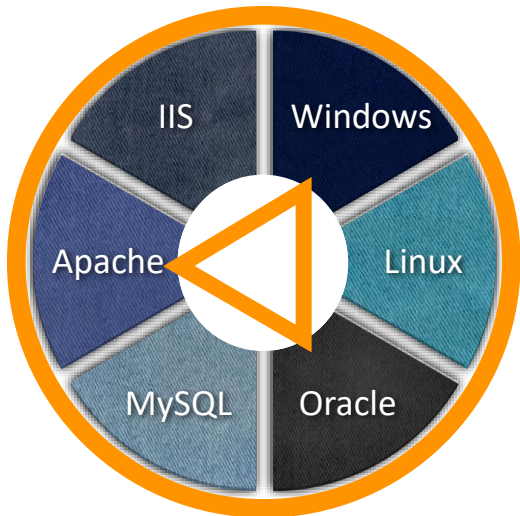
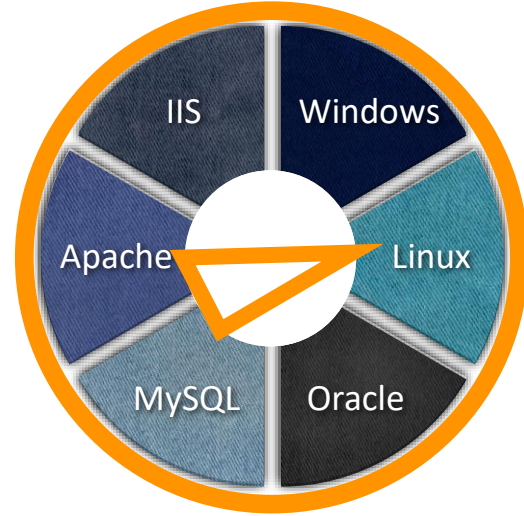
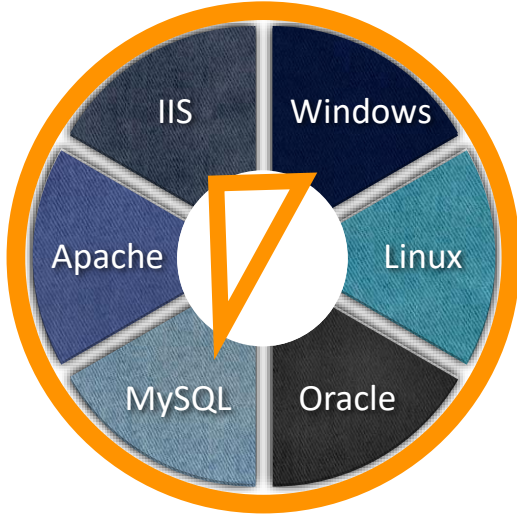
*Alle Paare* aus Kombinationen abdecken

wie etwa MySQL auf Windows und Linux

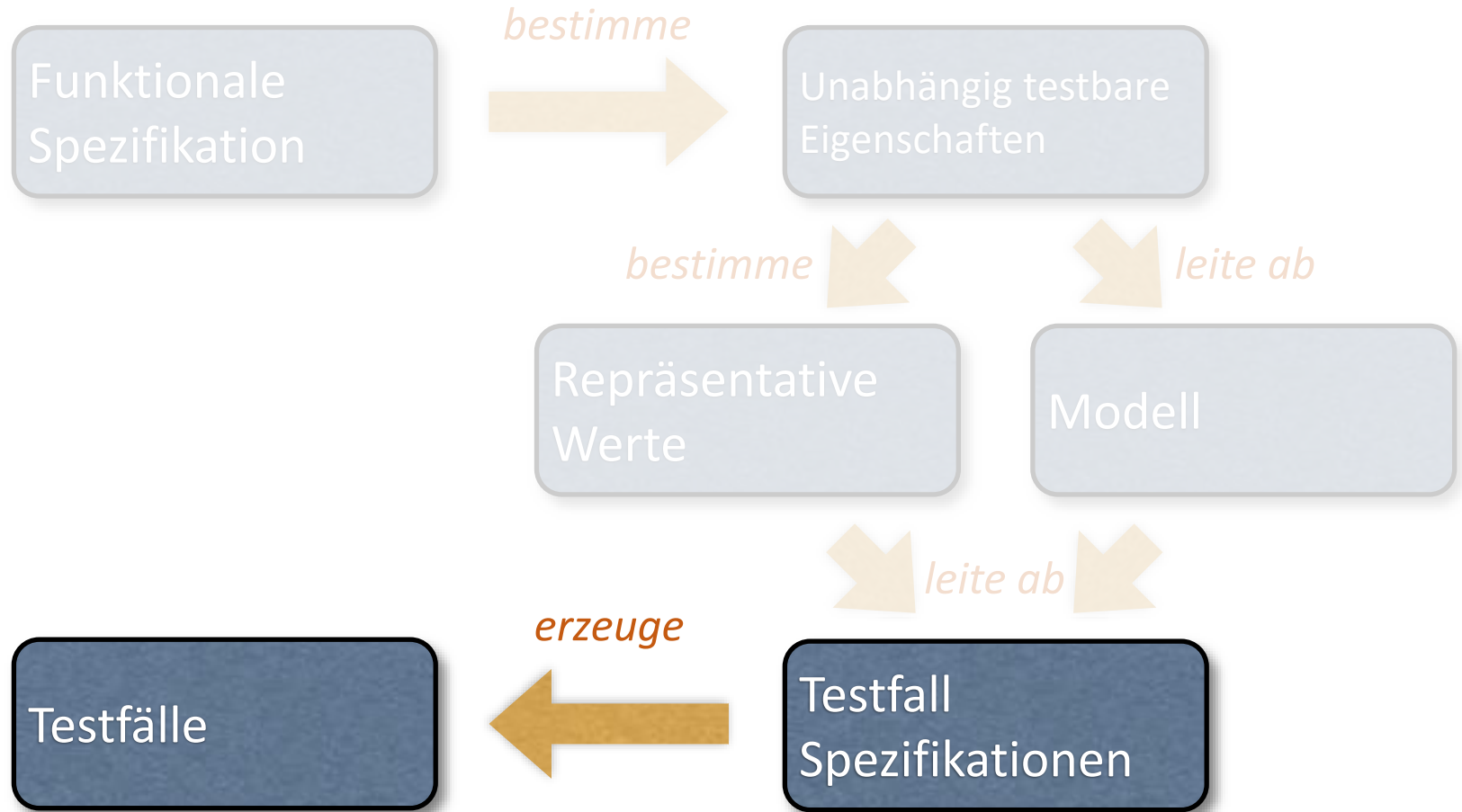
Kombinationen gewöhnlich *automatisch  
generiert*

und – hoffentlich – auch automatisch getestet

# Paarweises Testen



# Testfälle erzeugen



# Testfälle erzeugen

Testfälle in Code umsetzen

Benötigt *Testgerüste* –  
d.h. Testtreiber und Teststümpfe

