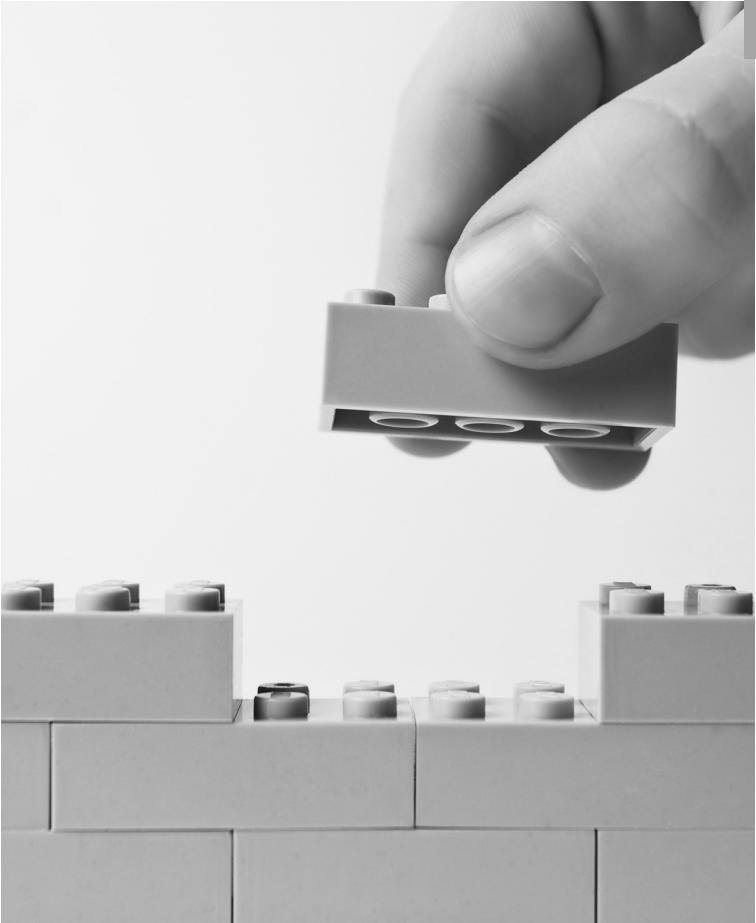


3 – Testanalyse und -entwurf

1. Testverfahren im Überblick
2. Black-Box-Testverfahren
3. White-Box-Testverfahren
4. Erfahrungsbasierte Testverfahren
5. Auf Zusammenarbeit basierende Testansätze



3.1 Testverfahren im Überblick

- FL-4.1.1 (K2) Sie können Black-Box-Testverfahren, White-Box-Testverfahren und erfahrungsbasierte Testverfahren unterscheiden

Zweck von Testverfahren

Testverfahren unterstützen den Tester ...

- bei der **Testanalyse** (was soll getestet werden) und beim **Testentwurf** (wie soll getestet werden).
- eine relativ kleine, aber **ausreichende Menge von Testfällen** systematisch zu entwickeln.
- bei der Definition von **Testbedingungen**.
- bei der Identifizierung von **Überdeckungselementen**.
- bei der Identifizierung von **Testdaten** während der Testanalyse und des Testentwurfs.

Weitere Informationen zu Testverfahren sind in der Norm ISO/IEC/IEEE 29119-4 zu finden.

Testverfahren



Black-Box-Testverfahren

(spezifikationsbasierte Verfahren)
basieren auf einer Analyse des spezifizierten Verhaltens des Testobjekts.
Testfälle werden unabhängig von der Implementierung der Software erstellt.



White-Box-Testverfahren

(strukturbasierte Verfahren)
basieren auf einer Analyse der internen Struktur und Verarbeitung des Testobjekts.
Testfälle können erst nach dem Entwurf oder der Implementierung des Testobjekts erstellt werden.



Erfahrungsbasierte Testverfahren

nutzen das Wissen und die Erfahrung von Testern für den Entwurf und die Implementierung von Testfällen.
Können Fehlerzustände aufdecken, die bei anderen Testverfahren möglicherweise übersehen werden.

Überdeckung

Überdeckung

Der Grad, zu dem bestimmte Überdeckungselemente von einer Testsuite ausgeführt wurden, ausgedrückt in Prozent.

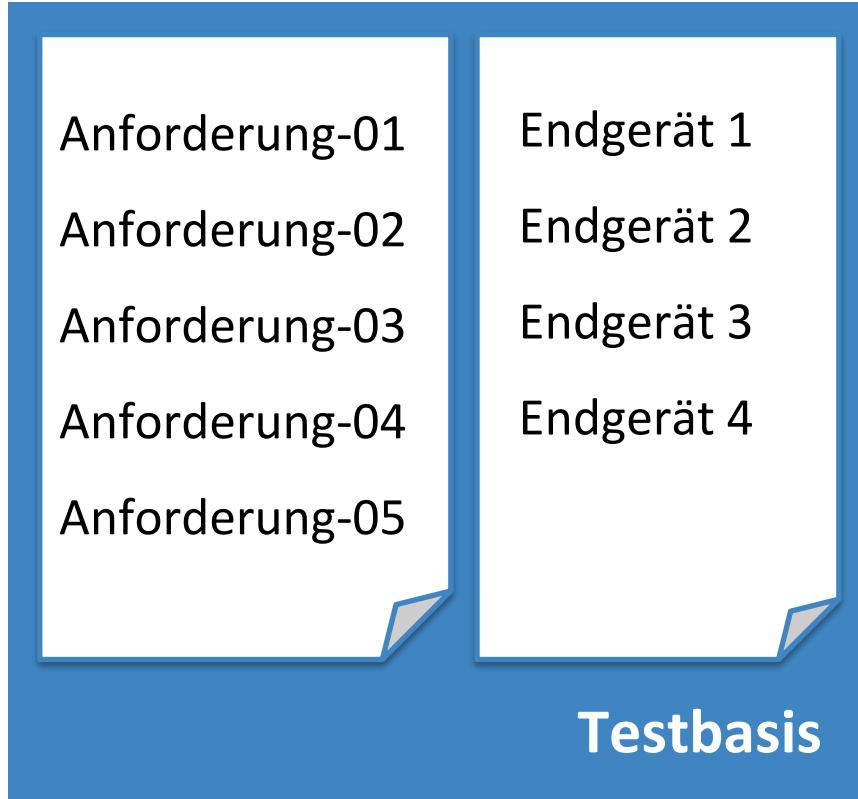
Überdeckungselemente

Eine Eigenschaft oder eine Kombination von Eigenschaften, die aus einer oder mehreren Testbedingungen unter Verwendung eines Testverfahrens abgeleitet wurde(n) .

$$\text{Überdeckung in \%} = \frac{[t] \text{ Anzahl der getesteten Überdeckungselemente}}{[n] \text{ Gesamtanzahl der identifizierten Überdeckungselemente}} * 100$$



Test einer mobilen Applikation



Überdeckungselemente

Für jedes Element der Testbasis
(Anforderungen sowie Endgeräte)
mindestens ein Testfall.



Testdurchführung



Ergebnis

- Sind alle spezifizierten Anforderungen erfüllt?
- Wurden Fehlerwirkungen in den unterstützten Endgeräten gefunden?



Schlüsselbegriffe – Testverfahren im Überblick

Testverfahren

Eine Vorgehensweise zum Definieren von Testbedingungen, Entwerfen von Testfällen und Spezifizieren von Testdaten.

Überdeckung

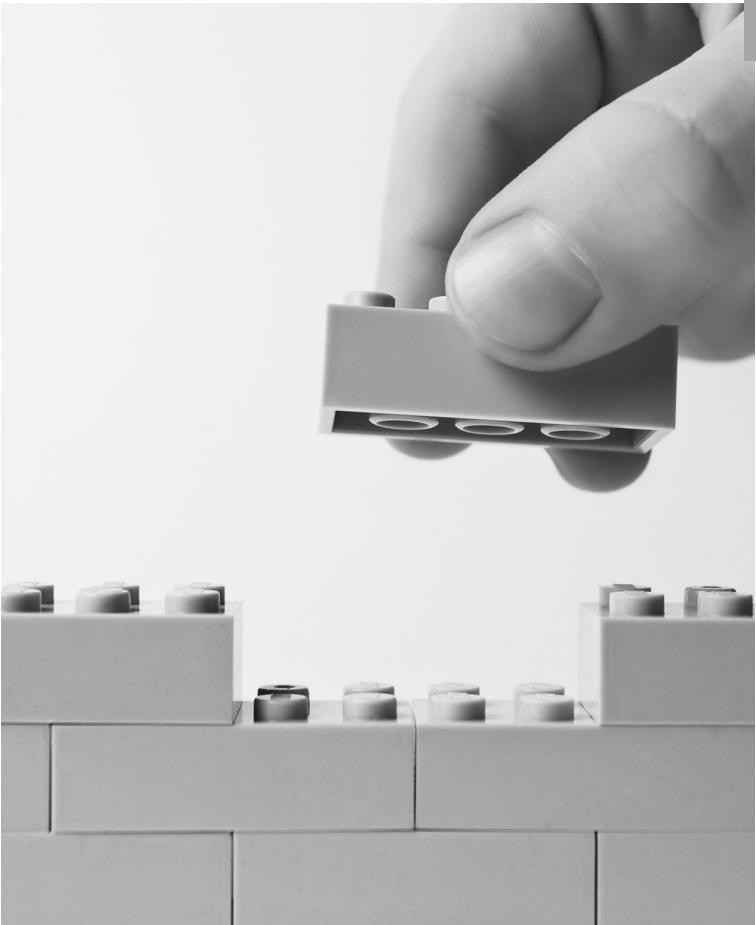
Der Grad, zu dem bestimmte Überdeckungselemente von einer Testsuite ausgeführt wurden, ausgedrückt in Prozent.

Überdeckungselement

Eine Eigenschaft oder eine Kombination von Eigenschaften, die aus einer oder mehreren Testbedingungen unter Verwendung eines Testverfahrens abgeleitet wurde(n).

3 – Testanalyse und -entwurf

1. Testverfahren im Überblick
2. Black-Box-Testverfahren
3. White-Box-Testverfahren
4. Erfahrungsbasierte Testverfahren
5. Auf Zusammenarbeit basierende Testansätze



3.2 Black-Box-Testverfahren

- FL-4.2.1 (K3) Sie können die Äquivalenzklassenbildung zur Ableitung von Testfällen anwenden
- FL-4.2.2 (K3) Sie können die Grenzwertanalyse zur Ableitung von Testfällen anwenden
- FL-4.2.3 (K3) Sie können den Entscheidungstabellentest zur Ableitung von Testfällen anwenden
- FL-4.2.4 (K3) Sie können den Zustandsübergangstest zur Ableitung von Testfällen anwenden

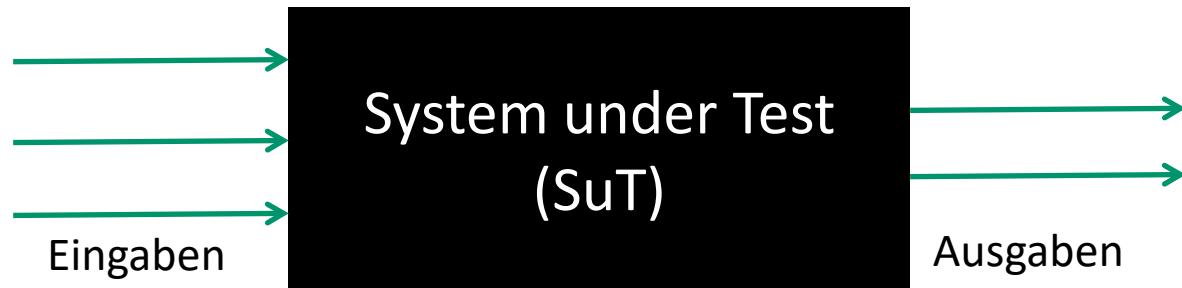
Black-Box-Testverfahren

Basieren auf einer **Analyse des spezifizierten Verhaltens** des Testobjekts,

z. B. **formale Anforderungsdokumente, Spezifikationen, Anwendungsfälle, User-Storys oder Geschäftsprozesse**

Die Testfälle werden unabhängig von der Implementierung der Software erstellt.

Folglich sind die Testfälle auch dann noch nützlich, wenn sich die Implementierung ändert, das geforderte Verhalten aber gleichbleibt.



Auch genannt: **Spezifikationsbasierte Verfahren**

Black-Box-Testverfahren

Übersicht

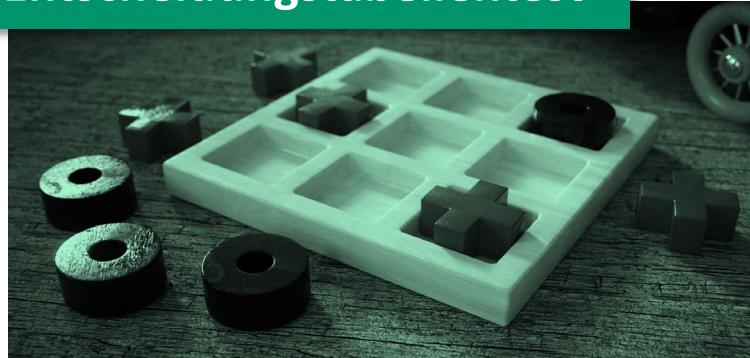
Äquivalenzklassenbildung



Grenzwertanalyse



Entscheidungstabellentest



Zustandsübergangstests



Anmerkung

Es gibt weitere Black-Box-Testverfahren.

Diese werden im ISTQB Certified Tester Advanced Level, Test Analyst besprochen.

Äquivalenzklassenbildung

Äquivalenzklassenbildung



Bei der Äquivalenzklassenbildung werden Daten in Bereiche (Klassen) unterteilt

Es wird davon ausgegangen, dass alle Elemente einer bestimmten Klasse vom Testobjekt auf die gleiche Weise verarbeitet werden.

→ **Ein Test für jede Klasse ist ausreichend.**

Denn eine Fehlerwirkung würde beim Test von jedem anderen beliebigen Wert aus derselben Klasse auch entdeckt werden.

Äquivalenzklassenbildung

Äquivalenzklassen können für jedes dem Testobjekt zugehörige Datenelement ermittelt werden, einschließlich

- Eingaben
- Ausgaben
- Konfigurationselemente
- interne Werte
- zeitbezogene Werte
- Schnittstellenparameter

Die Klassen können zusammenhängend oder einzeln, geordnet oder ungeordnet, endlich oder unendlich sein.

Die Klassen müssen nicht-leere Mengen sein und dürfen sich nicht überschneiden.

Die Äquivalenzklassenbildung sollte mit Sorgfalt vorgenommen werden, denn in der Praxis ist es oft kompliziert zu verstehen, wie das Testelement verschiedene Werte verarbeitet.

Gültige und ungültige Äquivalenzklassen

Gültige Äquivalenzklassen (gÄK)

Gültige Äquivalenzklassen (gÄK) beinhalten gültige Werte, die von der Komponente oder dem System **akzeptiert** werden sollten.

Ungültige Äquivalenzklassen (uÄK)

Ungültige Äquivalenzklassen (uÄK) beinhalten ungültige Werte, die von der Komponente oder dem System **abgelehnt** werden sollten.

Die Definitionen von gültigen und ungültigen Werten können je nach Team und Unternehmen variieren.

Zum Beispiel:

Gültige Werte werden vom Testobjekt verarbeitet oder die Spezifikation legt die Verarbeitung der Werte fest.

Ungültige Werte werden vom Testobjekt ignoriert oder zurückgewiesen, oder die Spezifikation des Testobjekts legt keine Verarbeitung fest.

Äquivalenzklassenbildung I



Datumsfeld „Monat Projektbeginn“

Spezifikation

- Die Eingabe erfolgt als ganze Zahl.
- Gültige Eingaben sind die Monate Januar (1) bis Dezember (12).
- Ab Oktober (10) wird der Tooltipp „Projektbeginn liegt im nächsten Geschäftsjahr!“ ausgegeben.
- Vor Oktober wird der Tooltipp „Projektbeginn liegt im aktuellen Geschäftsjahr“ ausgegeben.

Mit welchen konkreten Eingaben würden Sie das Datumsfeld testen?



Äquivalenzklassenbildung II

Datumsfeld „Monat Projektbeginn“

Basisklassen ermitteln

Für jede(n) Variable/Parameter:

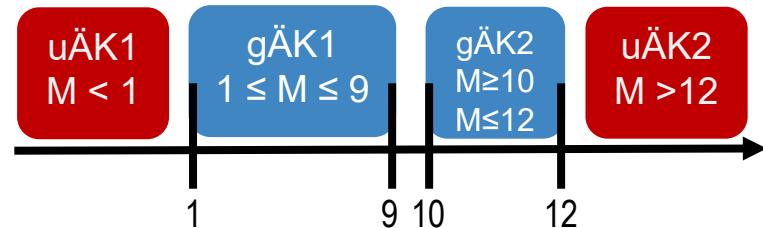
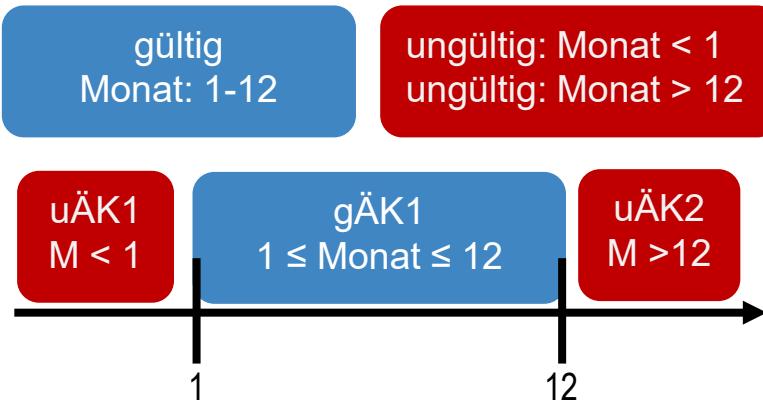
- Definitionsbereich ermitteln
- Klasse gültiger Werte bilden
- Klasse ungültiger Werte bilden

Klassen verfeinern

Für jede Äquivalenzklasse:

- Klassenelemente, die (vermutlich) unterschiedlich verarbeitet werden, einer neuen Klasse zuteilen.

Definitionsbereich
ganze Zahl





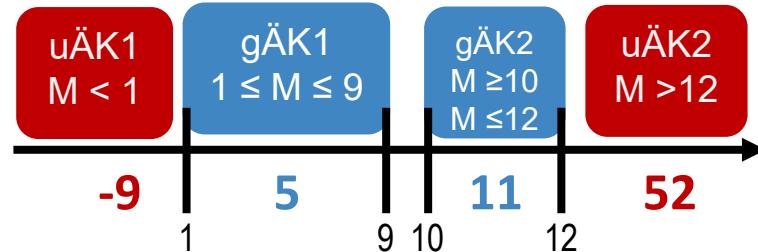
Äquivalenzklassenbildung III

Datumsfeld „Monat Projektbeginn“

Repräsentanten auswählen

Für jede Äquivalenzklasse:

- Ein repräsentatives Klassenelement als Wert für den Testfall auswählen.

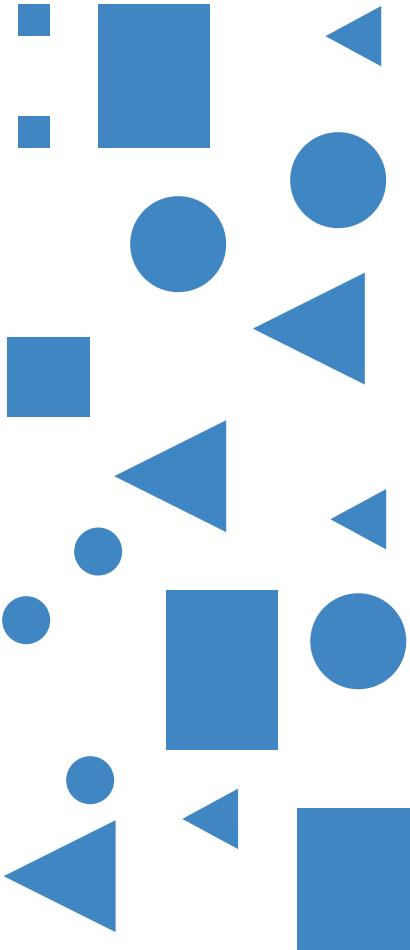


Ungültige Äquivalenzklassen

sollten individuell getestet werden, d.h. nur in Kombination mit anderen gültigen Äquivalenzklassen, um sicherzustellen, dass Fehlerwirkungen nicht maskiert bleiben.

- Fehlerwirkungen können maskiert bleiben, wenn mehrere Fehlerwirkungen zur gleichen Zeit auftreten, aber nur eine sichtbar ist, so dass die anderen Fehlerwirkungen unentdeckt bleiben.

Each-Choice-Überdeckung



Viele Testelemente umfassen mehrere Gruppen von Klassen (z. B. Testelemente mit mehr als einem Eingabeparameter).

Das bedeutet, dass ein Testfall Klassen aus verschiedenen Gruppen von Klassen abdeckt.

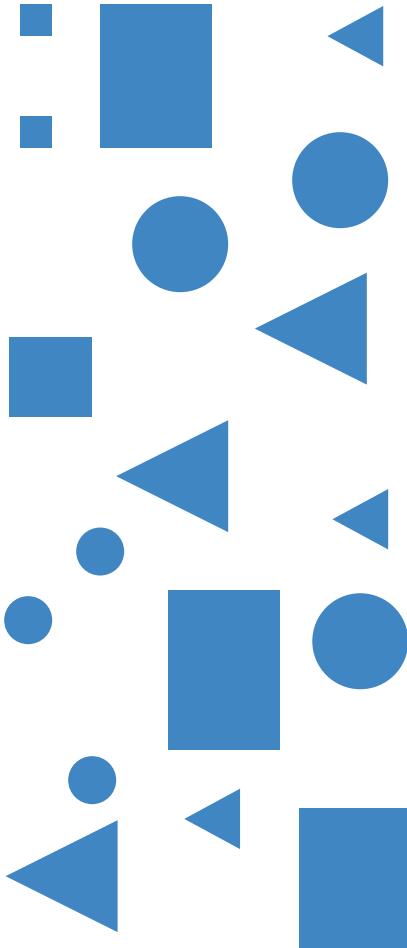
Die **Each-Choice-Überdeckung** ist das einfachste Überdeckungskriterium. Diese verlangt, dass jede Äquivalenzklasse **mindestens einmal** getestet wird.

Die Each-Choice-Überdeckung berücksichtigt **keine Kombinationen von Äquivalenzklassen**.

Für weitere Informationen zur Kombination von Äquivalenzklassen, siehe Certified Tester Advanced Level – Test Analyst.



Beispiel: Each-Choice-Überdeckung



Test eines Systems, das die Eigenschaften verschiedener geometrischer Objekte bestimmt.

Die Größe wird in ganzen Zahlen angegeben.

1-5 ist klein, 6-10 ist mittel, 11-20 ist groß, andere Größen sind ungültig.

Parameter

- Form: { Rechteck, Quadrat, Kreis, Dreieck }
- Größe: { <1, 1-5, 6-10, 11-20, >20 }

Testfall	Form	Größe
1	Rechteck	3
2	Quadrat	8
3	Kreis	15
4	Dreieck	klein / mittel / groß
5	Rechteck / Quadrat / Kreis / Dreieck	0
6	Rechteck / Quadrat / Kreis / Dreieck	30

Äquivalenzklassenbildung – Überdeckung

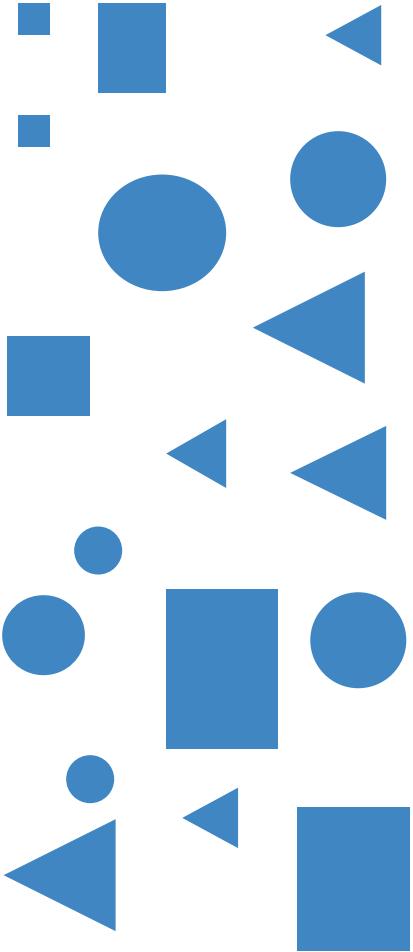
In der Äquivalenzklassenbildung sind die Überdeckungselemente die Äquivalenzklassen.

Um eine 100%ige Überdeckung zu erreichen, müssen die Testfälle alle identifizierten Klassen (einschließlich ungültiger Klassen) mindestens einmal ausführen.

$$\text{Überdeckung in \%} = \frac{[t] \text{ Anzahl der getesteten Äquivalenzklassen}}{[n] \text{ Gesamtanzahl der identifizierten Äquivalenzklassen}} * 100$$



Überdeckung messen

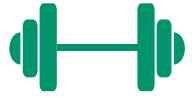


Parameter

- Form: { Rechteck, Quadrat, Kreis, Dreieck }
- Größe: { <1, 1-5, 6-10, 11-20, >20 }

Testfall	Form	Größe
1	Rechteck	3
Überdeckung	1 von 4	1 von 5
in Prozent	25%	20%

Testfall	Form	Größe
1	Rechteck	3
2	Quadrat	30
Überdeckung	2 von 4	2 von 5
in Prozent	50%	40%



3-1 Äquivalenzklassenbildung



Für den Test der Komponente „VSR-II EASY-FINANCE“ werden Testfälle mittels Äquivalenzklassenbildung ermittelt.

- Führen Sie eine Analyse der Testbasis durch.
- Identifizieren Sie Parameter und bilden dazu Äquivalenzklassen.
- Verwenden Sie das im Übungsheft enthaltene Schema.

Grenzwertanalyse

Grenzwertanalyse



- Basiert auf der Überprüfung der Grenzen von Äquivalenzklassen.
- Die Grenzwertanalyse kann nur für geordnete Klassen verwendet werden.
- Grenzwerte sind der Minimal- und Maximalwert einer Klasse.

Wenn zwei Elemente zur gleichen Klasse gehören, müssen bei der Grenzwertanalyse alle Elemente zwischen ihnen ebenfalls zu dieser Klasse gehören.

Grenzwertanalyse

Das Verhalten an den Grenzen von Äquivalenzklassen ist fehleranfälliger als das Verhalten innerhalb der Äquivalenzklassen.

Beispielsweise können spezifizierte und implementierte Grenzen

- auf Positionen oberhalb oder unterhalb ihrer vorgesehenen Positionen verschoben worden sein.
- gänzlich weggelassen worden sein.

Grenzwertanalyse und -tests decken fast alle derartigen Fehler auf, indem sie als Repräsentanten die unmittelbaren Werte der angrenzenden Klassen verwenden.

Wir unterscheiden zwei Versionen:

- 2-Wert-Grenzwertanalyse
- 3-Wert-Grenzwertanalyse

Grenzwertanalyse – Äquivalenzklassenbildung



- Die Eingabe erfolgt über ein Tastenfeld
- Das Eingabefeld akzeptiert eine einzige Ziffer
- Gültig ist die Eingabe von 1 bis 5



Gültige ÄK1: {1-5}

Ungültige ÄK1 (zu klein): {0} (Klasse mit einem Wert)

Ungültige ÄK2 (zu groß): {6-9}

Grenzwertanalyse – 2-Werte

Überdeckungselemente



2 Überdeckungselemente
je Grenzwert

2 Werte pro Grenze

0 , 1

5 , 6

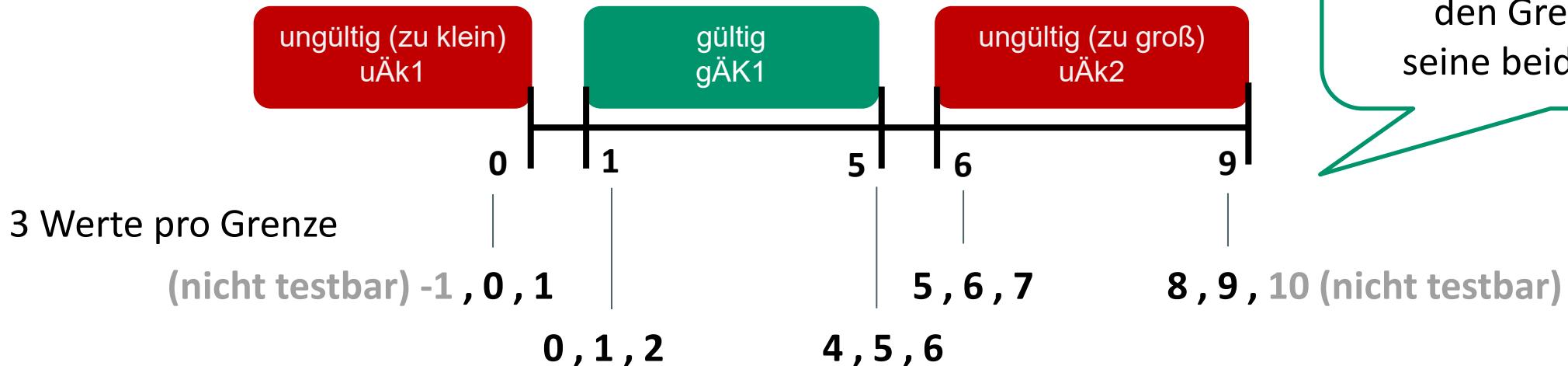
9 , 10 (nicht testbar)

Für eine 100%ige Überdeckung müssen die Testfälle alle Überdeckungselemente ausführen.

$$\text{Überdeckung in \%} = \frac{[t] \text{ Anzahl der ausgeführten Grenzwerte}}{[n] \text{ Gesamtzahl der identifizierten Grenzwerte}} * 100$$

Grenzwertanalyse – 3-Werte

Überdeckungselemente



3 Überdeckungselemente
je Grenzwert:
den Grenzwert und
seine beiden Nachbarn

Für eine 100%ige Überdeckung müssen die Testfälle alle Überdeckungselemente ausführen.

$$\text{Überdeckung in \%} = \frac{[t] \text{ Anzahl der ausgeführten Grenzwerte und ihrer Nachbarn}}{[n] \text{ Gesamtzahl der identifizierten Grenzwerte und ihrer Nachbarn}} * 100$$



Programmierfehler, der bei 2 Grenzwerten nicht gefunden wird



Implementierung korrekt

wenn ($x \geq 10$)

Implementierung fehlerhaft

wenn ($x == 10$)

Grenzwerte	9	10
Soll-Ergebnis	NOK	OK
Ist-Ergebnis	NOK	OK



Der vorhandene Fehler wird nicht entdeckt

Programmierfehler, der bei 3 Grenzwerten gefunden wird



Implementierung korrekt

wenn ($x \geq 10$)

Implementierung fehlerhaft

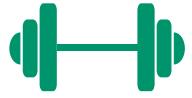
wenn ($x == 10$)

Grenzwerte	9	10	11
Soll-Ergebnis	NOK	OK	OK
Ist-Ergebnis	NOK	OK	NOK



Der vorhandene Fehler wird entdeckt

Die 3-Wert-Grenzwertanalyse ist strenger als die 2-Wert-Grenzwertanalyse, da sie Fehlerzustände aufdecken kann, die bei der 2-Wert-Grenzwertanalyse übersehen wurden.



3-2 Grenzwertanalyse



Ergänzend zur bereits durchgeführten Äquivalenzklassenbildung wird eine Grenzwertanalyse vorgenommen.

- Bestimmen Sie die Grenzwerte zu den in Übung 3-1 gefundenen Äquivalenzklassen.
- Verwenden Sie das im Übungsheft enthaltene Schema.

Entscheidungstabellentest

Entscheidungstabellentest

Entscheidungstabellen werden zum Testen der Umsetzung von Anforderungen verwendet, die angeben, wie verschiedene Kombinationen von Bedingungen zu unterschiedlichen Ergebnissen führen.

Entscheidungstabellen sind effektiv für die Aufzeichnung komplexer Logik, wie z. B. Geschäftsregeln.

Entscheidungstabellentest I

Eine Entscheidungstabelle besteht aus vier Quadranten.

Bedingungen	Regeln (J/N)
Aktionen	Aktionszeiger (X)

Bei der Erstellung werden die Bedingungen und die daraus resultierenden Aktionen des Systems definiert.
Diese bilden die Zeilen der Tabelle.
Jede Spalte entspricht einer Entscheidungsregel,
die eine eindeutige Kombination von Bedingungen zusammen mit den zugehörigen Aktionen definiert.



Entscheidungstabellentest II

Nr	Text	R ₁	R ₂	R ₃	R ₄
B ₁	Artikel im Lager	J	J	N	N
B ₂	Zahlungsverhalten ok	J	N	J	N
A ₁	Lieferung auf Rechnung	X			
A ₂	Lieferung nach Überweisung		X		
A ₃	Artikel nachbestellen			X	X
A ₄	Telefonischer Bescheid			X	
A ₅	E-Mail "Artikel wird nachbestellt"				X

Notation

Bedingungen

- J (Ja oder 1) → die Bedingung ist wahr
N (Nein oder 0) → die Bedingung ist falsch

Aktionen

- X (J, Ja oder 1) → die Aktion tritt ein
Leer (N, Nein oder 0) → die Aktion tritt nicht ein



Vollständige Entscheidungstabelle

Nr	Text	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆	R ₇	R ₈
B ₁	Artikelnummer gültig	J	J	J	J	N	N	N	N
B ₂	Artikel im Lager	J	J	N	N	J	J	N	N
B ₃	Zahlungsverhalten ok	J	N	J	N	J	N	J	N
A ₁	Lieferung auf Rechnung	X							
A ₂	Lieferung nach Überweisung		X						
A ₃	Artikel nachbestellen			X	X				
A ₄	Telefonischer Bescheid			X					
A ₅	E-Mail "Artikel wird nachbestellt"				X				
A ₆	E-Mail "Artikelnummer ungültig"					X	X	X	X

Eine vollständige Entscheidungstabelle deckt jede Kombination von Bedingungen ab.

Bei n Bedingungen gibt es genau 2^n Bedingungskombinationen (Spalten).



Vereinfachte / Minimierte Entscheidungstabelle I

Nr	Text	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆	R ₇	R ₈
B ₁	Artikelnummer gültig	J	J	J	J	N	N	N	N
B ₂	Artikel im Lager	J	J	N	N	J	J	N	N
B ₃	Zahlungsverhalten ok	J	N	J	N	J	N	J	N
A ₁	Lieferung auf Rechnung	X							
A ₂	Lieferung nach Überweisung		X						
A ₃	Artikel nachbestellen			X	X				
A ₄	Telefonischer Bescheid			X					
A ₅	E-Mail "Artikel wird nachbestellt"				X				
A ₆	E-Mail "Artikelnummer ungültig"					X	X	X	X

Die Tabelle kann minimiert werden, indem Spalten mit undurchführbaren Bedingungskombinationen gestrichen werden.

Beim vollständigen Ausfüllen einer Entscheidungstabelle können sich unlogische/unmögliche Kombinationen ergeben.

Ein Artikel mit einer ungültigen Artikelnummer kann sich nicht im Lager befinden.



Vereinfachte / Minimierte Entscheidungstabelle II

Nr	Text	R ₁	R ₂	R ₃	R ₄	R ₅
B ₁	Artikelnummer gültig	J	J	J	J	N
B ₂	Artikel im Lager	J	J	N	N	-
B ₃	Zahlungsverhalten ok	J	N	J	N	-
A ₁	Lieferung auf Rechnung	X				
A ₂	Lieferung nach Überweisung		X			
A ₃	Artikel nachbestellen			X	X	
A ₄	Telefonischer Bescheid			X		
A ₅	E-Mail "Artikel wird nachbestellt"				X	
A ₆	E-Mail "Artikelnummer ungültig"					X

Die Tabelle kann minimiert werden, indem Spalten, in denen einige Bedingungen keinen Einfluss auf das Ergebnis haben, in einer einzigen Spalte zusammengefasst werden

Vereinfachung

- (irrelevant) → der Wert der Bedingung ist für das Eintreten/Auslösen der Aktion irrelevant

Algorithmen zur Minimierung von Entscheidungstabellen sind nicht Gegenstand dieses Lehrplans.



Entscheidungstabelle mit erweiterter Eingabe

Nr	Text	R ₁	R ₂	R ₃	R ₄	R ₅
B ₁	Ampel ist eingeschaltet	J	J	J	J	N
B ₂	Farbe	grün	gelb	rot	rot - gelb	N/A
A ₁	Verkehr ist freigegeben	X				
A ₂	Vor der Kreuzung auf das nächste Zeichen warten		X			
A ₃	Halt vor Kreuzung			X		
A ₄	Auf Weiterfahrt vorbereiten				X	
A ₅	Verkehrszeichen beachten					X

Vereinfachung

- N/A (not applicable) → die Bedingung ist für eine bestimmte Regel undurchführbar

Entscheidungstabellen mit eingeschränkter Eingabe

Angabe der Bedingungen und Aktionen als

- Boolesche Werte (wahr oder falsch)

Entscheidungstabellen mit erweiterter Eingabe

Einige oder alle Bedingungen und Aktionen können auch mehrere Werte annehmen

- Zahlenbereiche
- Äquivalenzklassen
- Einzelwerte (z. B. rot, gelb, grün)

Entscheidungstabellentest – Überdeckung

Beim Entscheidungstabellentest sind Überdeckungselemente die Spalten, die ausführbare Kombinationen von Bedingungen enthalten.

Um eine 100%ige Überdeckung zu erreichen, müssen die Testfälle alle diese Spalten ausführen.

$$\text{Überdeckung in \%} = \frac{[t] \text{ Anzahl der ausgeführten Spalten}}{[n] \text{ Gesamtzahl der ausführbaren Spalten}} * 100$$

Überdeckung



Nr	Text	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆	R ₇	R ₈
B ₁	Artikelnummer gültig	J	J	J	J	N	N	N	N
B ₂	Artikel im Lager	J	J	N	N	J	J	N	N
B ₃	Zahlungsverhalten ok	J	N	J	N	J	N	J	N
A ₁	Lieferung auf Rechnung	X							
A ₂	Lieferung nach Überweisung		X						
A ₃	Artikel nachbestellen			X	X				
A ₄	Telefonischer Bescheid			X					
A ₅	E-Mail "Artikel wird nachbestellt"				X				
A ₆	E-Mail "Artikelnummer ungültig"					X	X	X	X

100% Überdeckung
erfordert 8 Testfälle

Nr	Text	R ₁	R ₂	R ₃	R ₄	R ₅
B ₁	Ampel ist eingeschaltet	J	J	J	J	N
B ₂	Farbe		grün	gelb	rot	rot - gelb
A ₁	Verkehr ist freigegeben	X				
A ₂	Vor der Kreuzung auf das nächste Zeichen warten			X		
A ₃	Halt vor Kreuzung				X	
A ₄	Auf Weiterfahrt vorbereiten					X
A ₅	Verkehrszeichen beachten					X

100% Überdeckung
erfordert 5 Testfälle

Entscheidungstabellentest – Stärken – Herausforderungen



Stärken

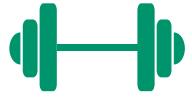
Entscheidungstabellen bieten einen systematischen Ansatz zur Identifizierung aller Kombinationen von Bedingungen.

- Kombinationen werden entdeckt, von denen einige andernfalls übersehen werden könnten.
- Lücken oder Widersprüche in den Anforderungen können gefunden werden.

Herausforderungen

Da die Anzahl der Regeln exponentiell mit den Bedingungen wächst, kann die Anwendung aller Entscheidungsregeln sehr zeitaufwändig sein.

- Um die Anzahl der zu testenden Regeln zu reduzieren, kann eine minimierte Entscheidungstabelle oder ein risikobasierter Ansatz verwendet werden.



3-3 Entscheidungstabellentest



Ein Programm zur Eröffnung von Girokonten soll getestet werden.

- Finden Sie Testfälle mit Hilfe einer Entscheidungstabelle.
- Verwenden Sie das im Übungsheft enthaltene Schema.

Zustandsübergangstest

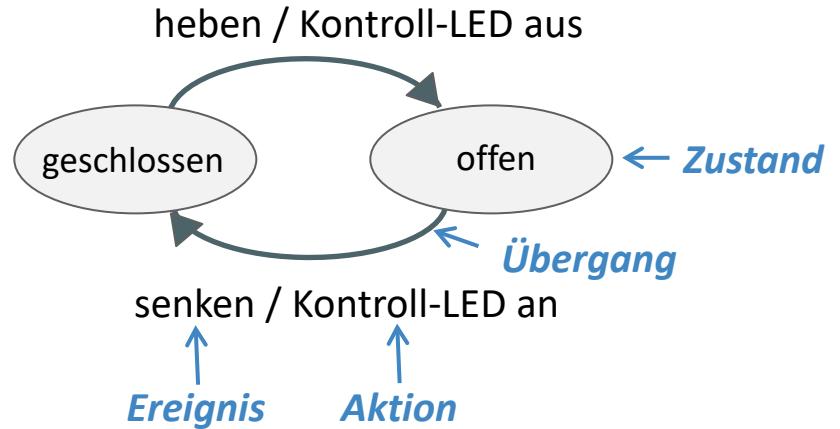


Einfacher Zustandsautomat

Beschränkter Bahnübergang



Zustandsautomat



Zustandsdiagramme sind gerichtete Graphen, die der graphischen Veranschaulichung dienen.

- Ein Übergang wird durch ein Ereignis ausgelöst, es wird davon ausgegangen, dass die Übergänge augenblicklich erfolgen
- Ein Übergang kann ggf. Aktionen / Ausgaben auslösen.

Zustandsübergangstest

Das Konzept eines Zustands ist abstrakt – es kann sich dabei um einige Zeilen Code oder gar um einen gesamten Geschäftsprozess handeln.

Berücksichtigung der aktuellen Gegebenheiten und der Vorgeschichte.

Die Zustandsänderung kann Reaktionen der Software zur Folge haben.

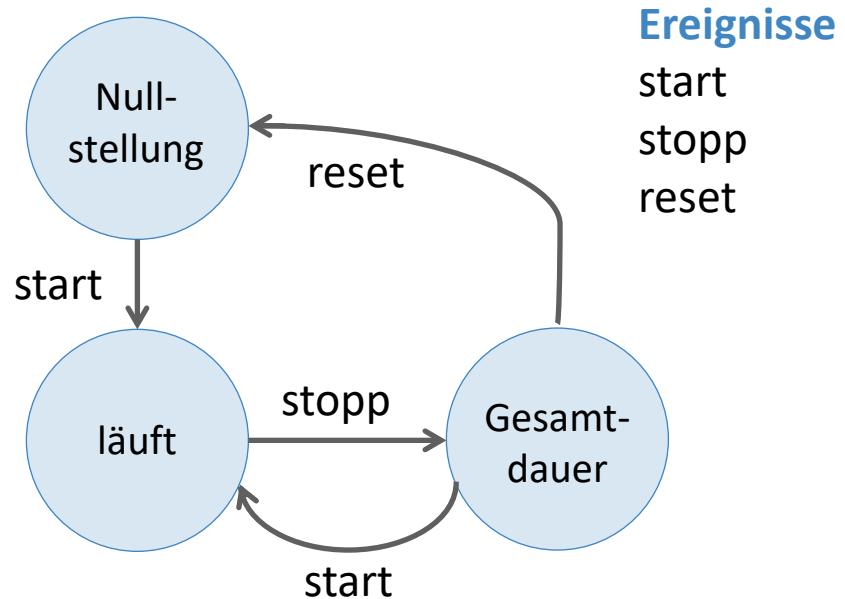
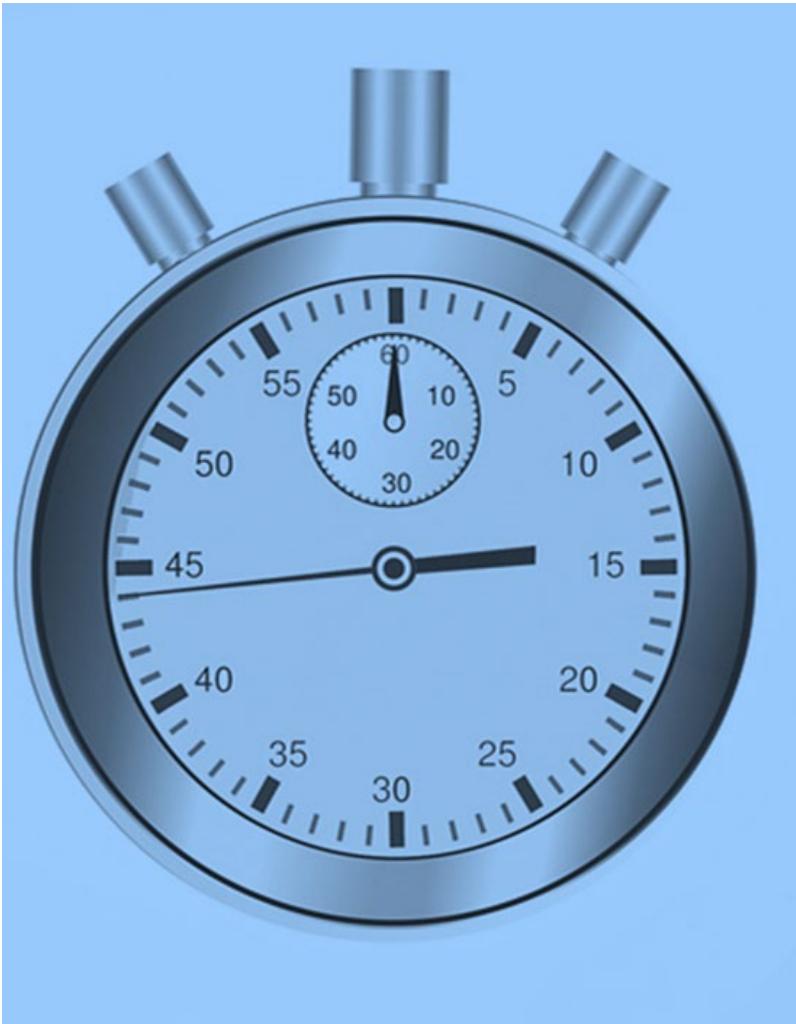
z. B. Ausgabe einer Berechnung oder einer Fehlermeldung.

Typische Anwendungsgebiete

- Eingebettete Systeme
(embedded systems)
- Menügeführte Anwendungen
- Bildschirmnavigation
- Modellerstellung eines Geschäftsszenarios



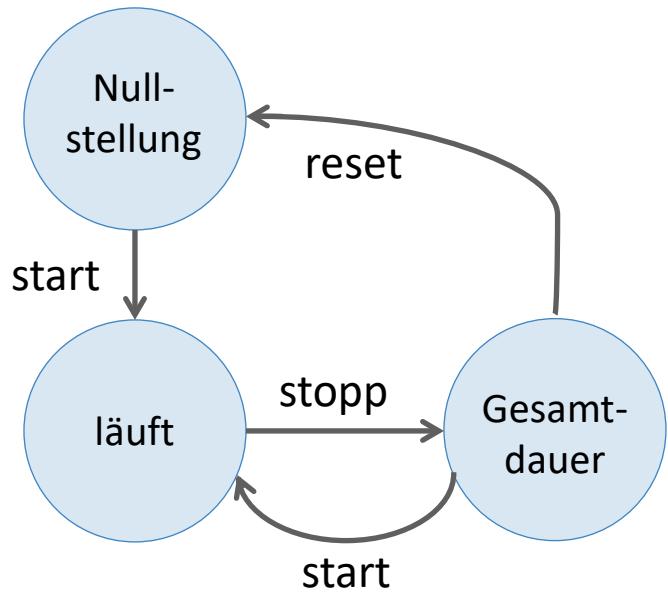
Zustandsautomat für eine Stoppuhr



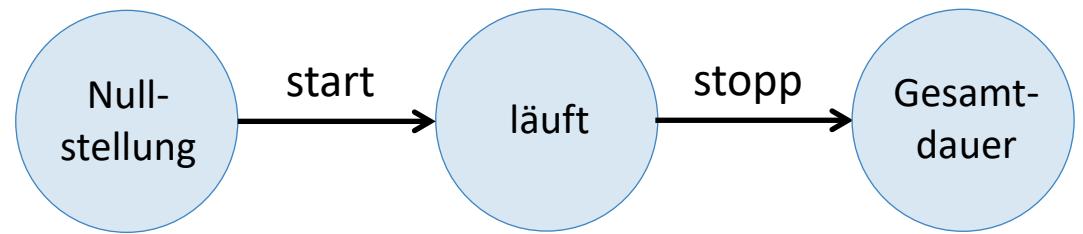


Testfälle für die Stoppuhr

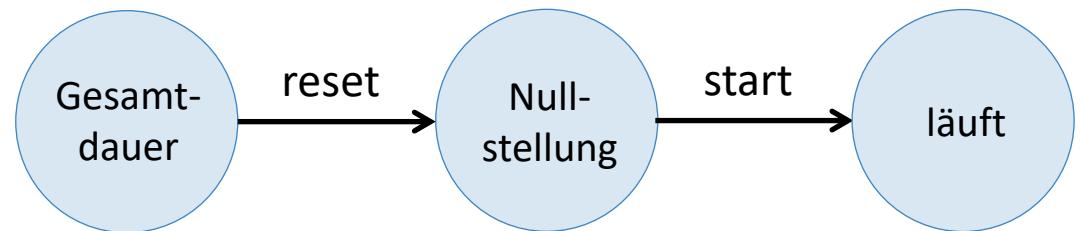
Testfälle sind **Folgen von Ereignissen**, die zu einer Abfolge von Zustandsänderungen (und ggf. Aktionen) führen. Ein Testfall kann mehrere Übergänge zwischen den Zuständen abdecken.



Oft verwendet man **Testfälle, die im Startzustand** des Zustandsautomaten **beginnen**, z. B.



Es können aber auch **beliebige Pfade** aus dem Zustandsautomat verwendet werden, z. B.



→ In diesem Fall ist der erste Zustand die Vorbedingung des Testfalls und der letzte Zustand die Nachbedingung des Testfalls.

Überdeckung aller Zustände I

z. B. Nullstellung, läuft, Gesamtdauer



$$\text{Überdeckung aller Zustände in \%} = \frac{[t] \text{ Anzahl der ausgeführten Zustände}}{[n] \text{ Gesamtzahl der Zustände}} * 100$$

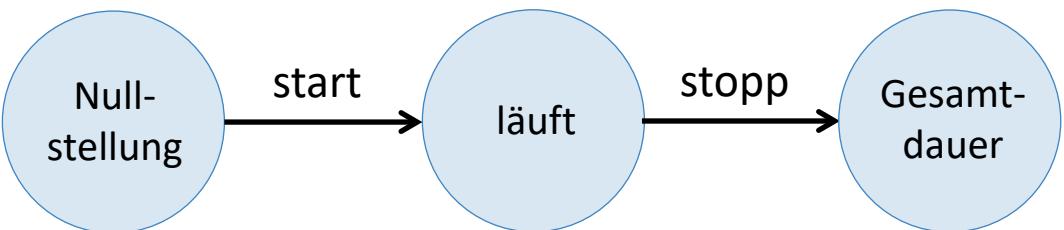
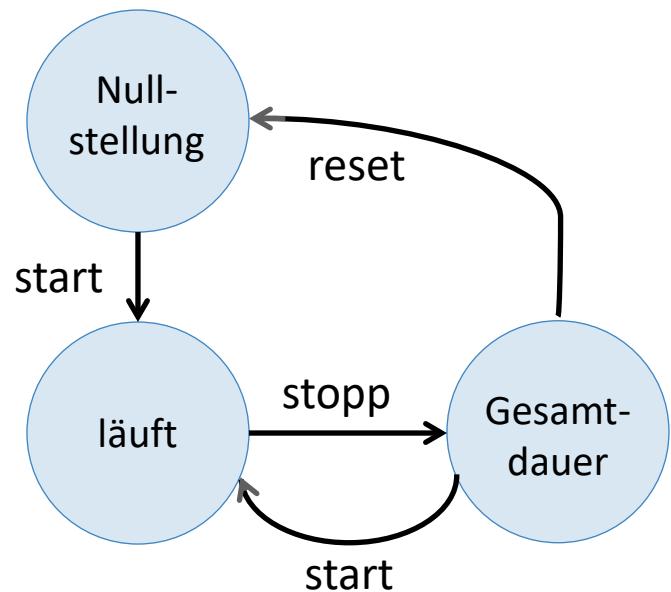
Die Überdeckung aller Zustände ist schwächer als die Überdeckung aller gültiger Übergänge,
da sie in der Regel erreicht werden kann, ohne alle Übergänge auszuführen.

Für weitere Informationen zu weiteren Überdeckungskriterien, siehe ISTQB-Lehrplan Advanced Level – Test Analyst.

Überdeckung aller Zustände II



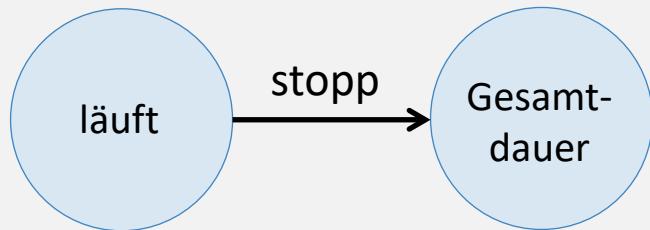
Finden Sie eine (minimale) Menge von Testfällen, so dass jeder Zustand des Zustandsautomaten mindestens einmal in einem Testfall vorkommt.



1 Testfall für 100% Zustandsüberdeckung

Überdeckung der gültigen Übergänge I

Überdeckung der einzelnen gültigen Übergänge (0-Switch-Überdeckung), z. B. läuft → Gesamtdauer



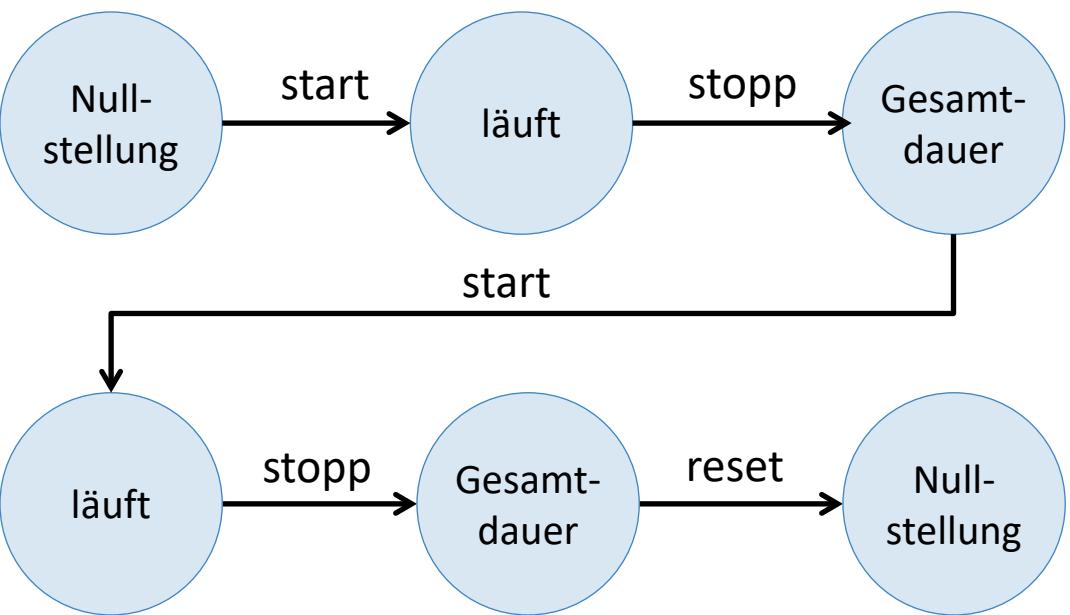
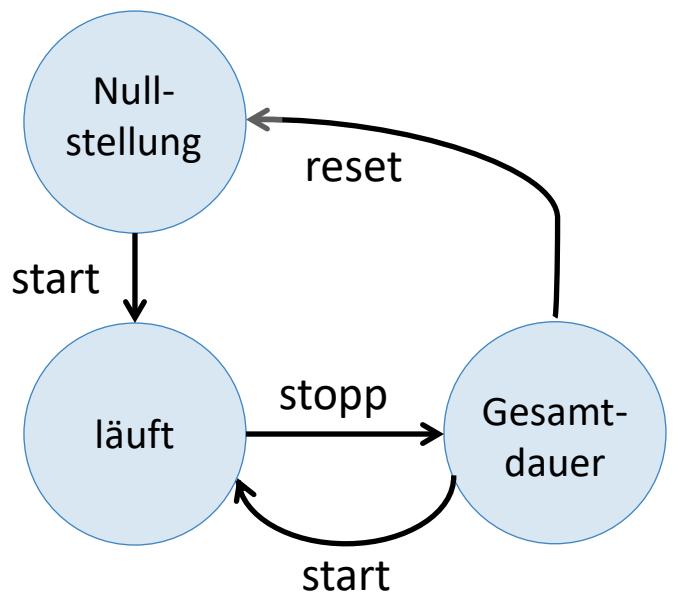
$$\text{Überdeckung der gültigen Übergänge in \%} = \frac{[t] \text{ Anzahl der ausgeführten gültigen Übergänge}}{[n] \text{ Gesamtzahl der gültigen Übergänge}} * 100$$

Die Überdeckung der gültigen Übergänge ist das am häufigsten verwendete Überdeckungskriterium.

Überdeckung der gültigen Übergänge II



Finden Sie die kleinstmögliche Anzahl von Testfällen, die jeden gültigen Übergang / Transition mindestens einmal durch einen Testfall abdeckt.



1 Testfall für 100% Überdeckung aller gültigen
Übergänge / Transitionen

Überdeckung aller Übergänge I

Überdeckung aller gültiger und ungültiger Übergänge



$$\text{Überdeckung aller Übergänge in \%} = \frac{[t] \text{ Anzahl der ausgeführten Übergänge}}{[n] \text{ Gesamtzahl der gültigen und ungültigen Übergänge}} * 100$$

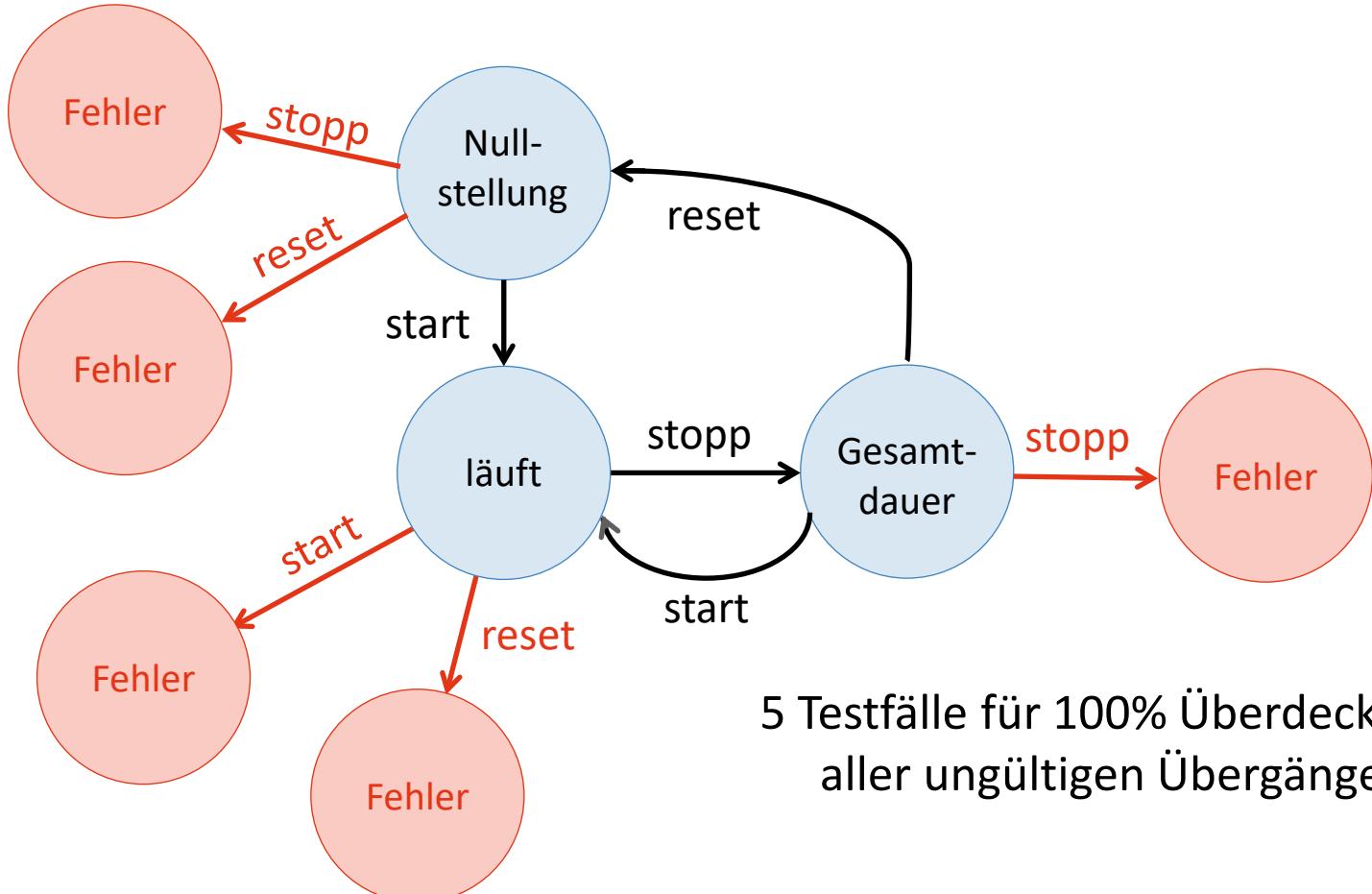
Jeden ungültigen Übergang mit einem einzelnen Testfall zu testen, hilft dabei Fehlermaskierung zu vermeiden.
 Fehlermaskierung ist eine Situation, in der ein Fehlerzustand die Entdeckung eines anderen verhindert.

Die vollständige Überdeckung aller Übergänge (gültig und ungültig) sollte eine Mindestanforderung für unternehmenskritische und sicherheitskritische Software sein.



Überdeckung der ungültigen Übergänge

Test auf Robustheit, Negativtests

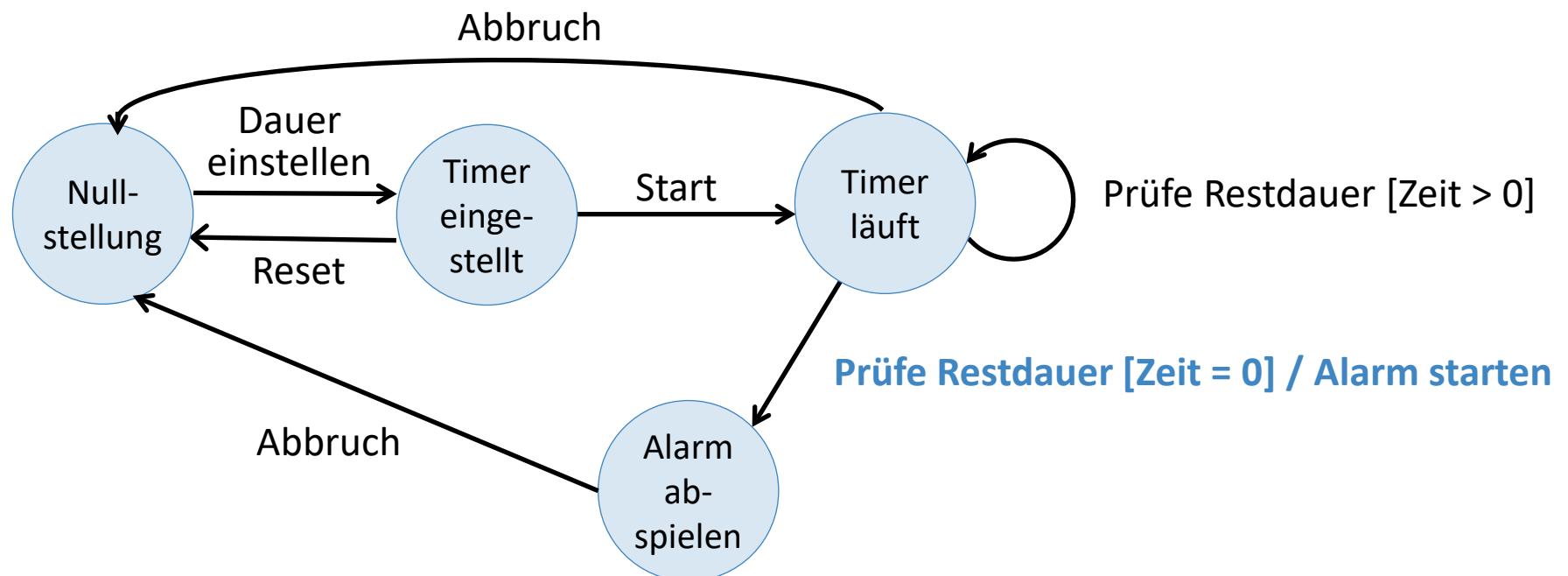




Wächterbedingung (Guard Condition)

Ein Übergang, der durch ein Ereignis ausgelöst wird kann zusätzlich durch eine Wächterbedingung (guard condition) eingeschränkt werden. Schreibweise: "Ereignis [Wächterbedingung] / Aktion"

Wächterbedingungen und Aktionen können weggelassen werden, wenn sie nicht existieren oder für den Tester irrelevant sind.

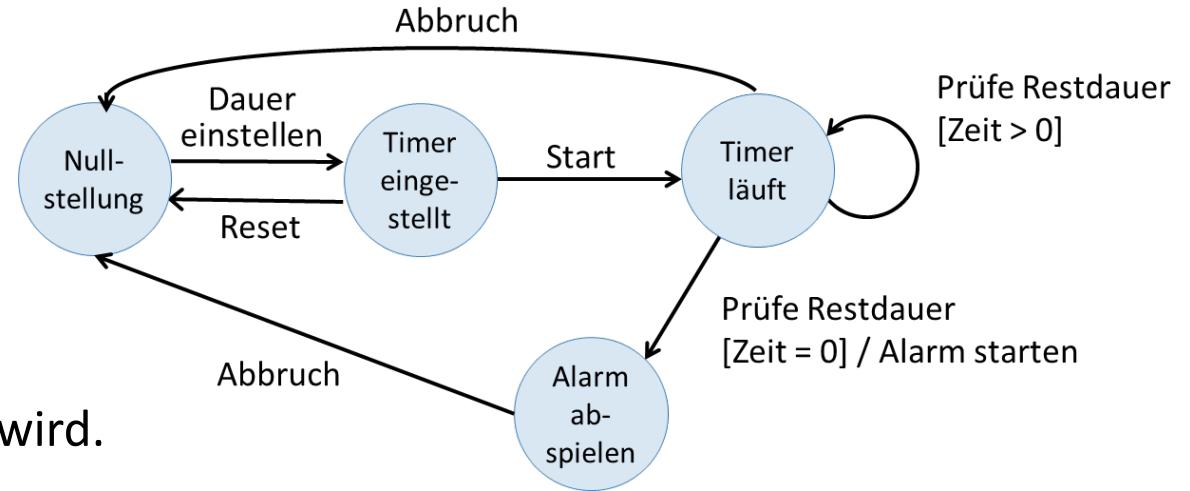




Zustandstabelle

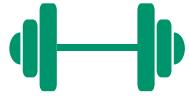
Zustandsautomaten können auch durch eine Zustandstabelle dargestellt werden.

Die Zustandstabelle zeigt für jeden Zustand, durch welche Ereignisse ein Zielzustand erreicht wird.



Zustand / Ereignis	Dauer einstellen	Reset	Start	Abbruch	Prüfe Restdauer [Zeit > 0]	Prüfe Restdauer [Zeit = 0] / Alarm starten
Nullstellung	Timer eingestellt					
Timer eingestellt		Nullstellung	Timer läuft			
Timer läuft				Nullstellung	Timer läuft	Alarm abspielen
Alarm abspielen				Nullstellung		

↑ ungültiger Übergang ↑ gültiger Übergang



3-4 Zustandsübergangstests



Leiten Sie anhand des im Übungsheft skizzierten Zustandsautomaten Testfälle zu einem vereinfachten Login-Vorgang ab.

- Ermitteln Sie die Zustände und die Ereignisse.
- Erstellen Sie eine Zustandsübergangstabelle auf Basis des Zustandsdiagramms.
- Ermitteln Sie Testfälle, um alle Zustände herbeizuführen und alle gültigen Übergänge auszuführen.



Schlüsselbegriffe – Black-Box-Testverfahren

Black-Box-Testverfahren

Ein Testverfahren, das auf einer Analyse der Spezifikation einer Komponente oder eines Systems basiert.

Grenzwertanalyse

Ein Black-Box-Testverfahren, bei dem die Testfälle unter Nutzung von Grenzwerten entworfen werden.

Entscheidungstabellentest

Ein Black-Box-Testverfahren, bei dem Testfälle im Hinblick auf die Ausführung von Kombinationen der Bedingungen und aus ihnen resultierender Aktionen einer Entscheidungstabelle entworfen werden.

Äquivalenzklassenbildung

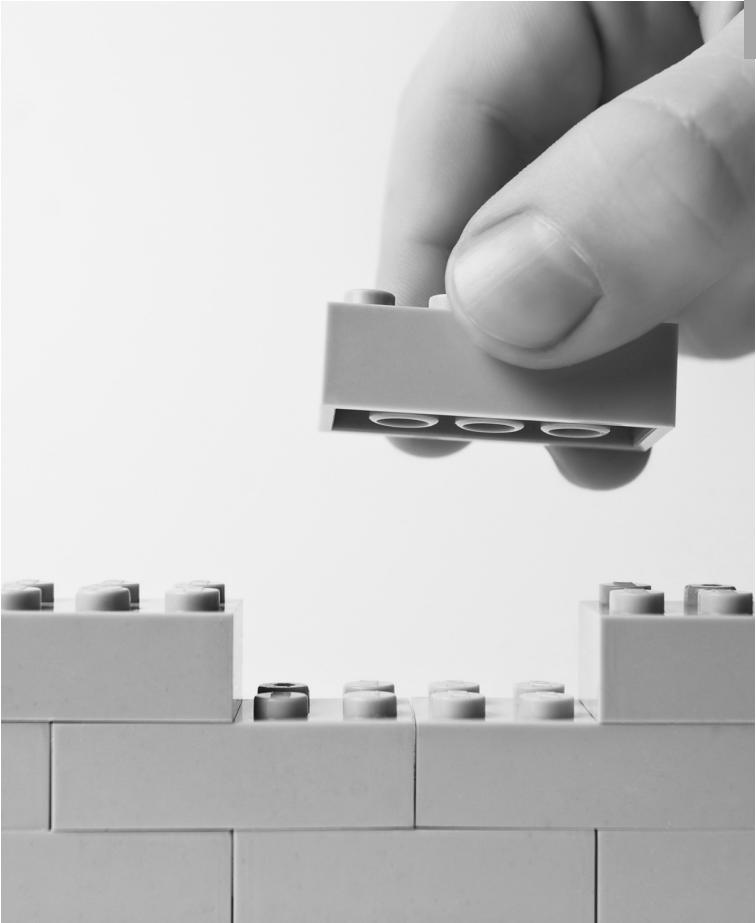
Ein Black-Box-Testverfahren, bei dem die Testbedingungen Äquivalenzklassen sind, und für jede Klasse ein repräsentatives Element ausgeführt wird.

Zustandsübergangstest

Ein Black-Box-Testverfahren, bei dem Testfälle entworfen werden, um Elemente eines Zustandsübergangsmodells auszuführen.

3 – Testanalyse und -entwurf

1. Testverfahren im Überblick
2. Black-Box-Testverfahren
3. White-Box-Testverfahren
4. Erfahrungsbasierte Testverfahren
5. Auf Zusammenarbeit basierende Testansätze



3.3 White-Box-Testverfahren

- FL-4.3.1 (K2) Sie können den Anweisungstest erklären
- FL-4.3.2 (K2) Sie können den Zweigtest erklären
- FL-4.3.3 (K2) Sie können den Wert des White-Box-Tests erklären

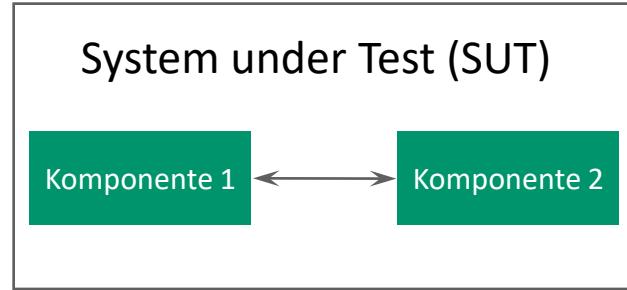
White-Box-Testverfahren

Basieren auf einer Analyse der internen Struktur und Verarbeitung des Testobjekts.

Testfälle können erst nach dem Entwurf oder der Implementierung des Testobjekts erstellt werden, da sie vom Entwurf der Software abhängig sind.

Auch genannt

- **strukturelle Verfahren**
- **strukturbasiertes Verfahren**



Für existierende Testfälle kann die Überdeckung des Testobjektes gemessen und darüber weitere Testfälle zur Erhöhung der Überdeckung systematisch abgeleitet werden.

White-Box-Testverfahren

Übersicht

Aufgrund ihrer Verbreitung und Einfachheit konzentriert sich dieses Seminar auf zwei Testverfahren.

Anweisungstest



Zweigtest



Weitere White-Box-Testverfahren

Weitere White-Box-Testverfahren, die in diesem Seminar nicht behandelt werden, sind:

- Gründlichere Testverfahren (z.B. MC/DC), die in einigen sicherheitskritischen, unternehmenskritischen oder hochgradig integrierten Umgebungen eingesetzt werden.
siehe ISTQB Certified Tester Advanced Level, Technischer Test Analyst.
- Testverfahren die in höheren Teststufen eingesetzt werden.
z. B. bei API-Test
- Testverfahren bei denen sich die Überdeckung nicht auf den Code bezieht.
z. B. Neuronen Überdeckung beim Testen von neuronalen Netzen

A photograph of a Texas Instruments TI-84 Plus CE calculator. The screen displays TI BASIC code and its output. The code is:

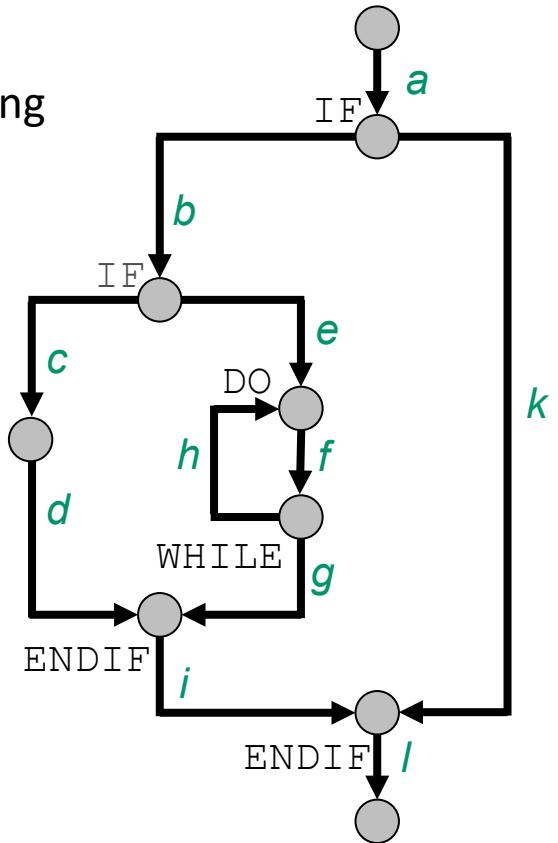
```
TI BASIC READY
>10 PRINT "HELLO WORLD!""
RUN
HELLO WORLD!
DONE
```

The background of the slide features a close-up, slightly blurred image of the calculator's keyboard and screen.

Anweisungstest

Anweisungstest

- Anweisung
- Zweig



Die Überdeckungselemente sind ausführbare Anweisungen.

Ziel: Mindestens einmalige Ausführung jeder Anweisung, bis eine akzeptable Überdeckung erreicht ist.

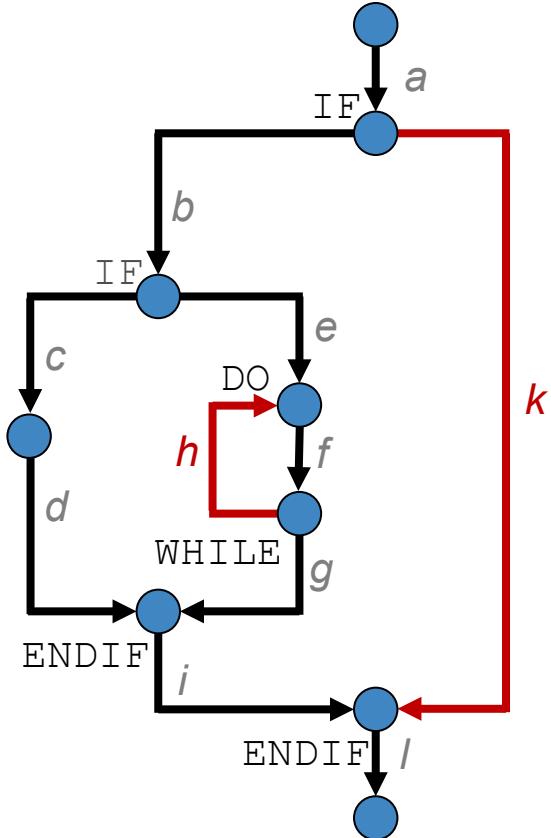
100% Anweisungsüberdeckung stellt sicher, dass alle Anweisungen im Code mindestens einmal ausgeführt wurden.

100% Anweisungsüberdeckung stellt nicht sicher, dass die gesamte Entscheidungslogik getestet wurde, da z. B. nicht alle Verzweigungen des Codes ausgeführt wurden.

$$\text{Anweisungsüberdeckung in \%} = \frac{[t] \text{ Anzahl der durch die Testfälle ausgeführten Anweisungen}}{[n] \text{ Gesamtzahl der ausführbaren Anweisungen im Code}} * 100$$



Anweisungsüberdeckung



100% - Anweisungsüberdeckung = {abcdil, abefgil}

Anweisungsüberdeckung wird oft als Minimum angesehen,
ist ein **schwaches Kriterium**, denn:

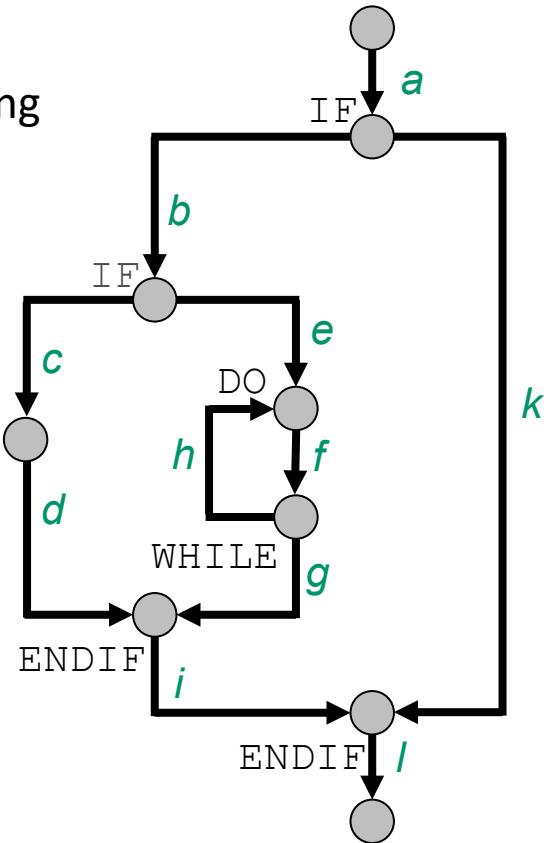
Leere Zweige (**k, h**) müssen nicht betreten werden,
→ falls dort eine Anweisung "fehlt",
wird dies nicht entdeckt.

Datenabhängige Fehler werden ggf. nicht erkannt
z. B. eine Division, die nur fehlschlägt, wenn
der Nenner auf „0“ gesetzt wird.

Zweigtest

Zweigtest

- Anweisung
- Zweig



Die Überdeckungselemente sind Zweige.

Ein Zweig ist ein Kontrollübergang zwischen zwei Knoten (Anweisungen) im Kontrollflussgraph.

Ziel: Mindestens einmalige Ausführung der Zweige im Code, bis eine akzeptable Überdeckung erreicht ist.

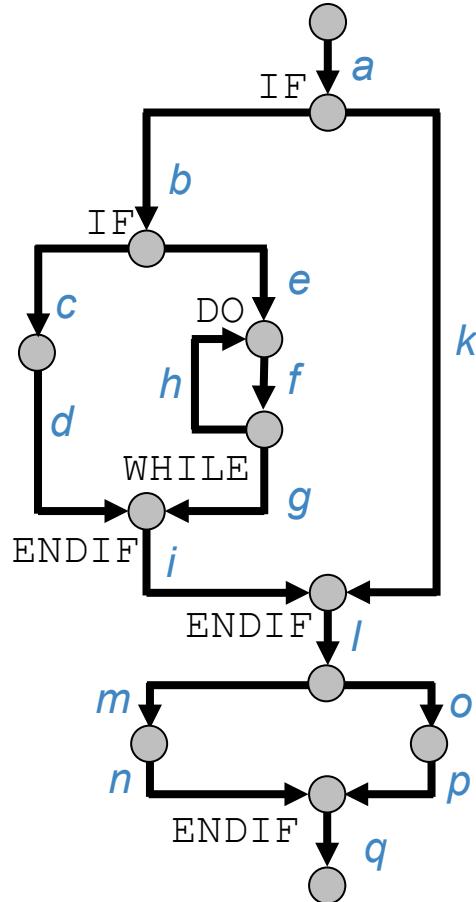
100% Zweigüberdeckung stellt sicher, dass alle Zweige des Codes (unbedingte und bedingte) durch Testfälle ausgeführt werden.

Das Erzielen von 100% Zweigüberdeckung garantiert 100% Anweisungsüberdeckung (aber nicht umgekehrt).

$$\text{Zweigüberdeckung in \%} = \frac{[t] \text{ Anzahl der durch die Testfälle ausgeführten Zweige}}{[n] \text{ Gesamtzahl der Zweige}} * 100$$



Zweigüberdeckung



100% - Zweigüberdeckung = {abcdilmnq, abefhfgilmnq, aklopq}

Einfach zu messendes, minimales Kriterium, unbeachtet bleiben ...

- **Einfluss von Zweigreihenfolge**

Fehlerzustände, die die Ausführung eines bestimmten Pfades in einem Code erfordern, werden ggf. nicht erkannt.

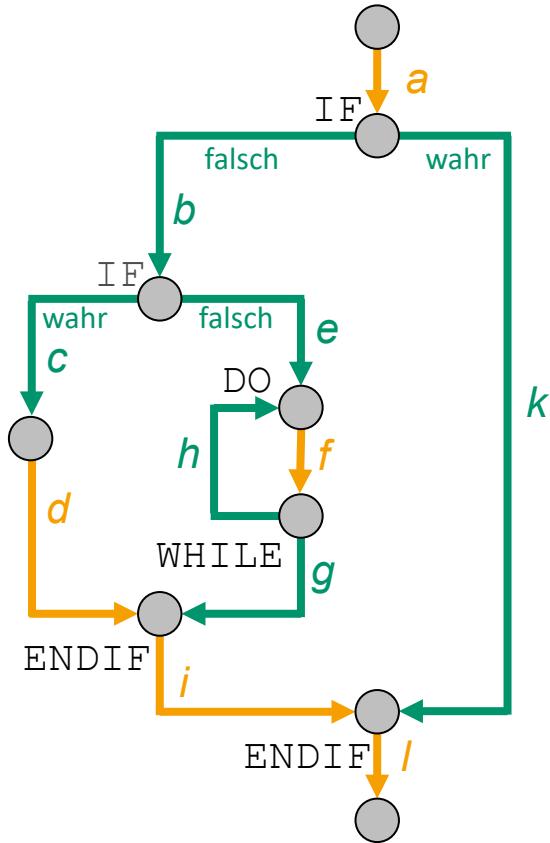
- **Schleifen**

Ein Fehlerzustand, der erst nach einer bestimmten Anzahl von Schleifendurchläufen auftritt, wird ggf. übersehen.

- **Bedingungen**

Fehlerzustände, die bei bestimmten Kombinationen von atomaren Teilbedingungen auftreten, werden ggf. übersehen.

Zweigtest – Kontrollübergänge



Jeder Kontrollübergang ist entweder bedingungslos oder bedingt.

Bedingungsloser Übergang (a, d, f, i, l)
bei geradlinigem Code

Bedingter Übergang (b, c, e, g, h, k)

bei einem bestimmten Entscheidungsergebnis

- **if....then** – wahres oder falsches Ergebnis
- **switch/case** – Anweisung mit mehreren Ergebnissen
- **Schleife** – Entscheidung über Fortsetzung oder Austritt

Der Wert des White-Box-Tests

White-Box-Test – Stärken

Die gesamte Implementierung wird berücksichtigt.

Fehlerzustände können unabhängig von der Softwarespezifikation entdeckt werden.
(Allerdings werden Abweichungen von Anforderungen möglicherweise nicht erkannt.)

White-Box-Tests können auch beim statischen Test eingesetzt werden.

z. B. beim Review von Code, Pseudocode oder Programmlogik, die mit einem Kontrollflussgraph modelliert werden kann.

White-Box-Tests liefern Informationen über fehlende Tests zu Erhöhung der Überdeckung.

Im Gegensatz zum Black-Box Test gibt es eine objektive Messgröße zur tatsächlichen Codeüberdeckung, das kann das Vertrauen in die Software stärken.

Wert des White-Box-Tests



Black-Box-Tests prüfen, ob eine Software den Anforderungen entspricht.

Bei einer Programmausführung kann es aber zu unvorhersehbaren Ereignissen kommen, die nicht zum normalen Programmablauf gehören.

Diese Ausnahmesituationen müssen durch Fehlerbehandlungen abgefangen und auch getestet werden.

- **Programminterne Fehlerbehandlungsroutinen sind in der Regel nicht vollständig in den Anforderungen beschrieben.**
- **Dafür müssen wir uns die Implementierung (den Sourcecode) ansehen und Testfälle ableiten, die jede dieser Fehlerbehandlungsroutinen im Sourcecode auslöst, damit wir diese Testen können.**



Schlüsselbegriffe – White-Box-Testverfahren

Zweigüberdeckung

Die Überdeckung von Zweigen in einem Kontrollflussgraphen.

Anweisungsüberdeckung

Die Überdeckung von ausführbaren Anweisungen.

White-Box-Testverfahren

Ein Testverfahren, das nur auf der inneren Struktur einer Komponente oder eines Systems basiert.

3 – Testanalyse und -entwurf

1. Testverfahren im Überblick
2. Black-Box-Testverfahren
3. White-Box-Testverfahren
4. Erfahrungsbasierte Testverfahren
5. Auf Zusammenarbeit basierende Testansätze



3.4 Erfahrungsbasierte Testverfahren

- FL-4.4.1 (K2) Sie können die intuitive Testfallermittlung erklären
- FL-4.4.2 (K2) Sie können den explorativen Test erklären
- FL-4.4.3 (K2) Sie können den checklistenbasierten Test erklären

Erfahrungsbasierte Testverfahren



Das Wissen und die Erfahrung von Testern wird beim Entwurf und bei der Implementierung von Testfällen effektiv genutzt.

Die Effektivität dieser Testverfahren hängt stark von den Kenntnissen der Tester ab.

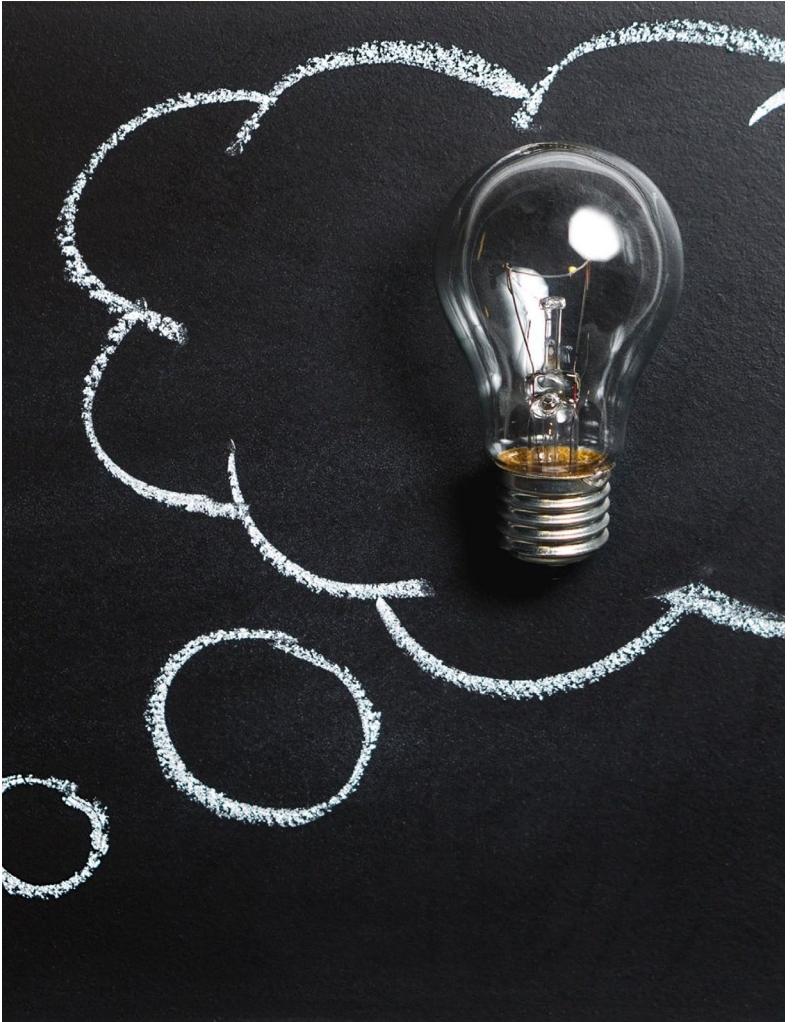
Erfahrungsbasierte Testverfahren werden oft mit Black-Box- und White-Box-Verfahren kombiniert. So können Fehlerzustände aufgedeckt werden, die andere Testverfahren möglicherweise übersehen.

Erfahrungsbasierte Testverfahren sind:

- Intuitive Testfallermittlung
- Explorativer Test
- Checklistenbasierter Test

Intuitive Testfallermittlung

Intuitive Testfallermittlung



Ein Testverfahren zur Vorhersage des Auftretens von Fehlhandlungen, Fehlerzuständen und Fehlerwirkungen, die auf dem Wissen des Testers basiert.

Bei der intuitiven Testfallermittlung listet ein Tester alle möglichen Fehlerzustände oder fehlerträchtigen Situationen auf, die aus seiner Praxis bekannt sind, z. B.

- Wie hat die Anwendung in der Vergangenheit funktioniert?
- Welche Arten von Fehlhandlungen werden üblicherweise von Entwicklern gemacht und welche Arten von Fehlerzuständen resultieren daraus?
- Bekannte Fehlerwirkungen in anderen Anwendungen?

Anschließend werden Testfälle erstellt, um diese Fehlerzustände zu finden.

Intuitive Testfallermittlung – Typische Fehler

Im Allgemeinen können sich **Fehlhandlungen, Fehlerzustände und Fehlerwirkungen** auf Folgendes beziehen:

- Eingabe (z. B. korrekte Eingabe nicht akzeptiert, falsche oder fehlende Parameter)
- Ausgabe (z. B. falsches Format, falsches Ergebnis)
- Logik (z. B. fehlende Fälle, falscher Operator)
- Berechnung (z. B. falscher Operand, falsche Berechnung)
- Schnittstellen (z. B. fehlende Parameterübereinstimmung, inkompatible Typen)
- Daten (z. B. falsche Initialisierung, falscher Typ)



Typische Fehler

Auswertung von Datumsformaten: Alterseingabe mit Geburtsdatum = 01.12.2005

- Auswertung deutsches Format: geboren am 01. Dezember 2005
- Auswertung amerikanisches Format: geboren am January 12, 2005

Ausgabe eines Betrages in verschiedenen Währungen

- Deutsches Format: 1.000,00 Euro
- Amerikanisches Format: 1,000.00 Dollar

Verwendung von falschen oder fehlenden Maßeinheiten in internationalen Projekten

- Verwechslung von Newton/s und lb/s (Mars Climate Orbiter)

Alter oder Jahreszahl hat zu wenige Stellen

- Jahr 2000 Problem
- Geburtsjahr = „23“. Ist die Person im Jahr 2025 eher 2 Jahre oder 102 Jahre alt?

Fehlerangriff



Fehlerangriff

Fehlerangriffe sind eine Möglichkeit der intuitiven Testfallermittlungen.

Tester erstellen oder übernehmen eine Liste möglicher Fehlhandlungen, Fehlerzustände und Fehlerwirkungen und entwerfen Testfälle, um die mit den Fehlhandlungen verbundenen Fehlerzustände zu identifizieren, die Fehlerzustände aufdecken oder die Fehlerwirkungen verursachen.

Grundlage für die Erstellung der Listen:

- Erfahrungswerte
- Daten über Fehlerzustände und Fehlerwirkungen
- allgemeines Wissen darüber, warum Software fehlschlägt

Explorativer Test



Exploratives Testen

Informelles Testverfahren, das den Einsatz anderer Black-Box-, White-Box- oder erfahrungsbasierter Testverfahren beinhalten kann.

Beim explorativen Test werden **Tests gleichzeitig entworfen, ausgeführt und bewertet**, während der Tester mehr über das Testobjekt erfährt.

Zusätzlich wird das Testobjekt mit gezielten Tests gründlicher erforscht und weitere Tests für ungetestete Bereiche erstellt.

- Geeigneter Ansatz, wenn wenig oder unzureichende Dokumentation verfügbar ist.
- Beliebter Ansatz bei Zeitdruck.
- Geeignet zur Ergänzung formal ermittelter Tests.
- Ist effektiver, wenn der Tester erfahren ist und über Fachkenntnisse verfügt.
- Erfordert grundlegende Kompetenzen, wie analytische Fähigkeiten, Neugier und Kreativität.

Sitzungsbasiertes Testen

Um den explorativen Test zu strukturieren, wird das Testen oft als **sitzungsbasierter Test** (session-based testing) durchgeführt.

- Tests werden innerhalb eines definierten Zeitfensters (60-120 Min.) durchgeführt.
- Eine Test-Charta mit Testzielen (z. B. Testbedingungen, die aus der Analyse übernommen wurden) steuert den Test.
- Überdeckungselemente werden während der Testsitzung identifiziert und ausgeführt.
- Sitzungsblätter (Session-Sheets) dokumentieren die ausgeführten Testschritte und die Ergebnisse.
- Nachbesprechung der Testergebnisse mit allen Beteiligten.

Test-Charta



Test-Charta = Testziel + Testideen

Test-Charta	Beispiel (nach Jon Bach):
=	
Testziel	Untersuche Funktionalität <...>.
+ Testideen	Beachte dabei insbesondere Risikoaspekte, Aussagen in der Spezifikation und Stabilität. Habe ein besonderes Auge auf das Zeitverhalten (z. B. Verzögerungen nach „Speichern“).

Umfrage



- Überlegen Sie gemeinsam mit dem Trainer mögliche Testziele für den Virtual-Showroom II.
- Welche Testideen sind in den Testzielen enthalten?
- Wie würden Sie die Testziele priorisieren?

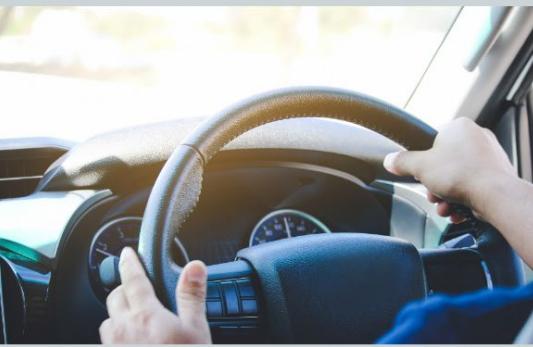


Explorativer Test – VSR-II

The screenshot shows a car configuration interface. On the left, a list of options is shown with checkboxes and their corresponding prices:

Option	Preis
Autopilot Don'tCare	2.900,00 €
Winterreifen (anstatt Sommerreifen)	210,00 €
Navigationssystem	1.490,00 €
Fußmatten	46,00 €
Lederlenkrad	360,00 €
Sound System OverDrive	470,00 €
Sportfederung	900,00 €
Intelligentes Lichtsystem	2.000,00 €

Below the list, it says "Preis Zusatzausstattung: 3.974,60 €". At the bottom, there's a "Konfigurationsname: Testfahrzeug 1" field and a "Gesamtpreis: 24.674,60 €" field.



Interessenten können ein Auto frei konfigurieren, bekommen eine Preisübersicht



Dokumentation der Sitzung und Erfassung der Aufwände (%) mit STB-Metrik
Setup: Zeitaufwand für die Testvorbereitungen
Test: Zeitaufwand für Testentwurf und Testdurchführung
Bugs: Zeitaufwand für Fehlerlokalisierung und Dokumentation

Glossarvideo – Exploratives Testen



IMBUS SOFTWARETEST

GLOSSAR VIDEOS

Checklistenbasierter Test

Checklistenbasierter Test



Tester entwerfen, implementieren und führen Tests aus, um Testbedingungen aus einer Checkliste abzudecken.

Grundlage für die Erstellung von Checklisten:

- Erfahrungen
- Wissen darüber, was für den Benutzer wichtig ist
- Verständnis darüber, warum und wie Software fehlgeschlagen ist

Das sollte nicht in eine Checkliste:

- Elemente, die automatisch geprüft werden können
- Elemente, die sich besser als Eingangskriterien oder Endekriterien eignen
- Elemente, die zu allgemein sind

Checklisten – Inhalt

Die Elemente von Checklisten ...

- sind häufig in Form von Fragen formuliert.
- sollten ermöglichen, jedes Element einzeln und direkt zu überprüfen.
- können sich auf Anforderungen, Eigenschaften grafischer Benutzeroberflächen, Qualitätsmerkmale oder andere Formen von Testbedingungen beziehen.

Checklisten können verschiedene Testarten (funktional und nicht-funktional) unterstützen.

Bei fehlenden Testfällen kann checklistenbasiertes Testen Richtlinien liefern und ein bestimmtes Maß an Konsistenz liefern.

Überdeckung

Da Checklisten auf einer eher allgemeineren Ebene formuliert sind, ist Variabilität beim Testen zu erwarten. Diese Variabilität führt zu einer höheren Überdeckung, aber auch zu geringerer Wiederholbarkeit.

10 Heuristiken für Gebrauchstauglichkeitstests



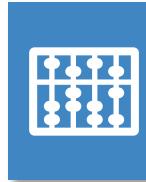
Systemzustand: Sichtbarkeit des Systemzustands



Ästhetik: Ästhetisches und minimalistisches Design



Übereinstimmung: Übereinstimmung zwischen dem System und der Wirklichkeit



Konsistenz: Konsistenz und Standards



Wiedererkennung: Wiedererkennen statt Erinnern



Hilfestellung: Hilfestellung beim Erkennen, Bewerten und Beheben von Fehlern



Flexibilität: Flexibilität und Effizienz der Anwendung



Fehlerprävention: Fehlervermeidung



Kontrolle & Freiheit: Nutzerkontrolle und Freiheit für den Benutzer



Dokumentation: Hilfe und Dokumentation

Checklisten – Was ist zu beachten?



Wie alle anderen Testmittel, sollten Checklisten regelmäßig aktualisiert werden.

Entwickler lernen, dieselben Fehlhandlungen zu vermeiden.

- Einige Einträge in der Checkliste können im Laufe der Zeit an Effektivität verlieren.

Es werden neue Fehlerzustände mit hohem Fehlerschweregrad gefunden.

- Möglicherweise müssen der Checkliste neue Einträge hinzugefügt werden.

Die Checkliste sollte nicht zu lang werden.

Schlüsselbegriffe – Erfahrungsbasierte Testverfahren



checklistenbasierter Test

Ein erfahrungsbasiertes Testverfahren, bei dem die Testfälle entworfen werden, um Elemente einer Checkliste auszuführen.

erfahrungsbasierte Testverfahren

Ein Testverfahren, das auf der Erfahrung, dem Wissen und der Intuition der Tester basiert.

explorativer Test

Ein Testansatz, bei dem die Tester auf der Grundlage ihres Wissens, der Erkundung des Testobjekts und der Ergebnisse früherer Tests dynamisch Tests entwerfen und durchführen.

intuitive Testfallermittlung

Ein Testverfahren, bei dem Tests auf Basis des Wissens der Tester über frühere Fehlerwirkungen oder auf Basis von allgemeinem Wissen über Fehlerauswirkungen abgeleitet werden.



Testverfahren – Gegenüberstellung I

	Black-Box- Testverfahren	White-Box- Testverfahren	Erfahrungsbasierte Testverfahren
Auch genannt	spezifikationsbasierte Verfahren	strukturbasierte Verfahren	
Testfall- ableitung	Test, bei dem Testfälle und Testdaten aus der funktionalen oder auch nicht-funktionalen Spezifikation (auch modellhaft) des Testobjekts abgeleitet werden ohne die (Programm-) Struktur auszuwerten.	Tests, bei denen Testfälle und Testdaten aus der Struktur des Testobjekts (z. B. Design oder Code) abgeleitet werden.	<p>Testfälle basierend auf ...</p> <ul style="list-style-type: none">▪ Erfahrung von Testern, Entwicklern, Anwendern und anderen über das Testobjekt und seine Umgebungen.▪ Erfahrungen, wo Fehler in der Vergangenheit aufgetreten sind oder Vermutungen, wo Fehler in Zukunft wahrscheinlich auftreten werden.

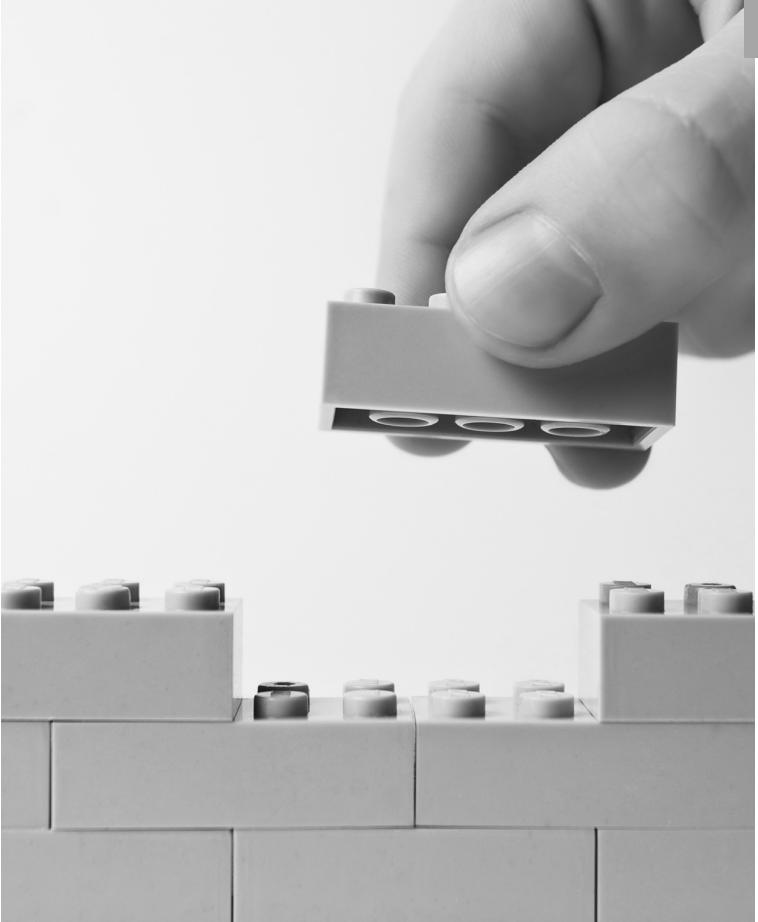


Testverfahren – Gegenüberstellung II

	Black-Box-Testverfahren	White-Box-Testverfahren	Erfahrungsbasierte Testverfahren
Gemeinsame Merkmale	<ul style="list-style-type: none">▪ Modelle, ob formal oder nicht formal, werden zur Spezifikation des zu lösenden Problems, der Software oder ihrer Komponente herangezogen.▪ Testfälle können systematisch von diesen Modellen abgeleitet werden.	<ul style="list-style-type: none">▪ Informationen über den Aufbau der Software werden für die Ableitung von Testfällen verwendet (z. B. Code und Informationen des Detailentwurfs).▪ Die Überdeckung der Software kann für vorhandene Testfälle gemessen werden.▪ Weitere Testfälle können zur Erhöhung der Überdeckung systematisch abgeleitet werden.	<ul style="list-style-type: none">▪ Das Wissen und die Erfahrung von Menschen werden zur Ableitung der Testfälle genutzt.▪ Das Wissen von Testern, Entwicklern, Anwendern und Betroffenen über die Software, ihre Verwendung und ihre Umgebung, wahrscheinliche Fehlerzustände und deren Verteilung ist eine Informationsquelle.
Testverfahren	<ul style="list-style-type: none">▪ Äquivalenzklassenbildung▪ Grenzwertanalyse▪ Entscheidungstabellentest▪ Zustandsübergangstest	<ul style="list-style-type: none">▪ Anweisungstest▪ Zweigtest	<ul style="list-style-type: none">▪ Intuitive Testfallermittlung▪ Explorativer Test▪ Checklistenbasierter Test

3 – Testanalyse und -entwurf

1. Testverfahren im Überblick
2. Black-Box-Testverfahren
3. White-Box-Testverfahren
4. Erfahrungsbasierte Testverfahren
5. Auf Zusammenarbeit basierende Testansätze



3.5. Auf Zusammenarbeit basierende Testansätze

- **FL-4.5.1 (K2)** Sie können das Schreiben von User-Storys in Zusammenarbeit mit Entwicklern und Fachvertretern erklären
- **FL-4.5.2 (K2)** Sie können die verschiedenen Möglichkeiten zum Schreiben von Akzeptanzkriterien einordnen
- **FL-4.5.3 (K3)** Sie können abnahmetestgetriebene Entwicklung (ATDD) zur Ableitung von Testfällen anwenden

Gemeinsames Schreiben von User-Storys

User-Story – die 3 C's

Jeffries 2000



Card

Karte

Medium, das eine User-Story beschreibt

- Karteikarte
- Eintrag auf einem elektronischen Board

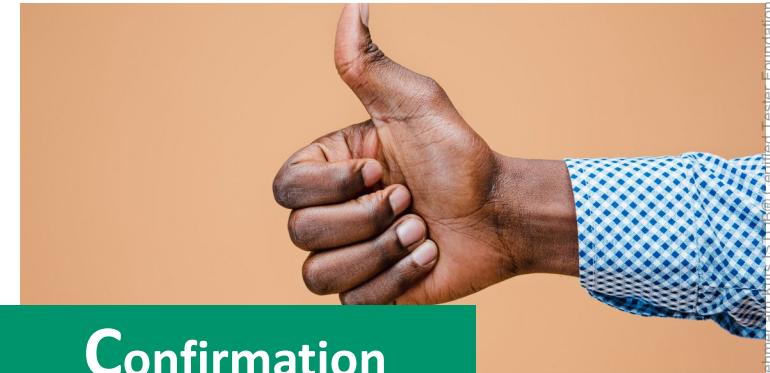


Conversation

Konversation

klärt, wie die Software genutzt werden soll

- dokumentiert
- mündlich



Confirmation

Bestätigung

Akzeptanzkriterien

- positive und negative Tests

Gemeinsames Schreiben von User-Storys

Eine User-Story beschreibt eine Funktion, die für einen Benutzer oder Käufer eines Systems oder einer Software nützlich sein wird.

Format einer User-Story:

"Als [Rolle] möchte ich, dass [das Ziel erreicht wird], so dass ich [resultierender Geschäftswert für die Rolle]",
gefolgt von den Akzeptanzkriterien.

Beispiel:

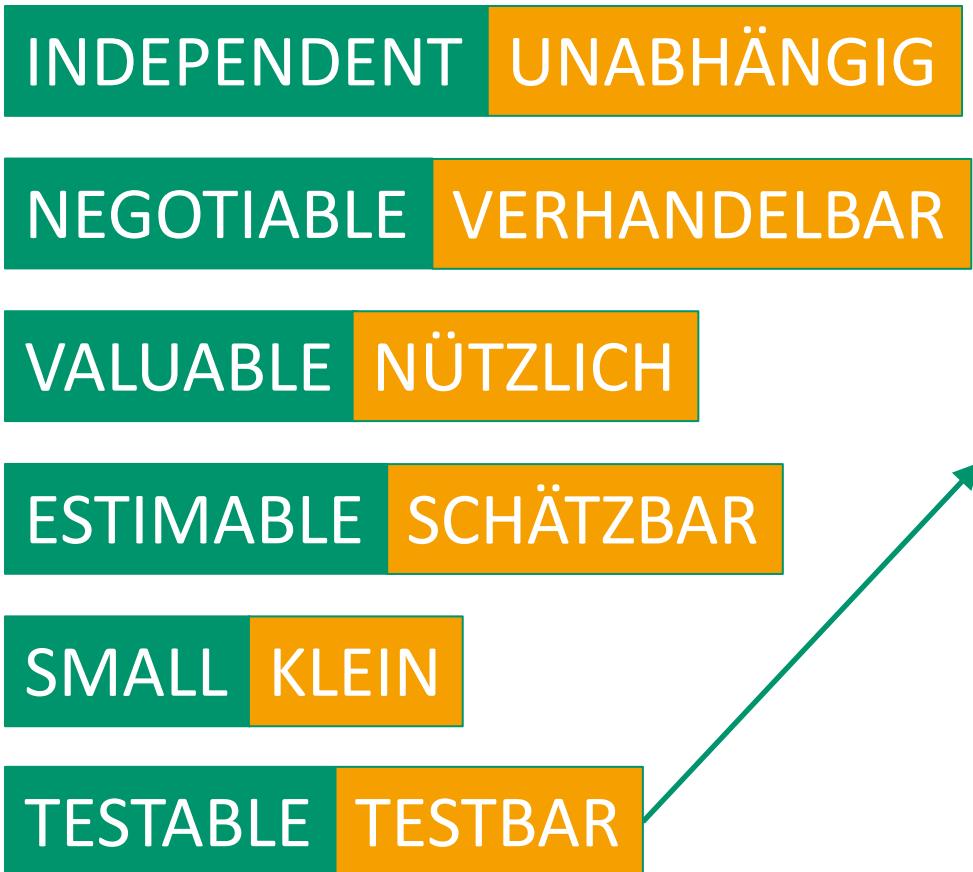
"Als Autohändler möchte ich Rabatte anbieten können, um mit potenziellen Käufern verhandeln zu können."

Andere Attribute: Beschreibung, Priorität, Schätzung, Risiko

Eigenschaften guter User-Storys

Wake 2003

I
N
V
E
S
T



Wenn ein Stakeholder nicht weiß, wie er eine User-Story testen soll, kann dies darauf hindeuten, dass ...

- die User-Story nicht klar genug ist
- die User-Story nichts Nützliches für den Stakeholder widerspiegelt
- der Stakeholder nur Hilfe beim Testen benötigt

Gemeinsames Schreiben von User-Storys

Ziele:

knapp – ausreichend – notwendig

Verfahren: z. B.

- Brainstorming
- Mind-Mapping

Die Zusammenarbeit ermöglicht es dem Team, eine gemeinsame Vision von dem zu erhalten, was geliefert werden soll, indem drei Perspektiven berücksichtigt werden:

Fachlichkeit, Entwicklung und Testen



VSR-II User-Storys



User-Story: 1 Darlehensbetrag berechnen

Als Kunde möchte ich eine Anzahlung machen, um den Darlehensbetrag zu verringern.

Der Darlehensbetrag wird wie folgt berechnet:

$$\text{Darlehensbetrag} = \text{Kaufpreis} - \text{Anzahlung}$$

Der Kaufpreis (in € und Cent) wird aus dem konfigurierten Fahrzeug übernommen.

Die Anzahlung muss mindestens 1,00 € betragen und darf den Kaufpreis des Fahrzeugs - 1,00 € nicht übersteigen.

Akzeptanzkriterien:

- 1 Für Anzahlung wird eine Dezimalzahl mit maximal zwei Nachkommastellen akzeptiert.
- 2 Bei Eingabe eines gültigen Wertes für Anzahlung wird der Darlehensbetrag berechnet.
- 3 Wird für Anzahlung kein Wert eingegeben, wird die Fehlermeldung „Geben Sie bitte einen Betrag ein.“ unter dem Eingabefeld ausgegeben.
- 4 Bei Eingabe eines ungültigen Wertes für Anzahlung wird die Fehlermeldung „Bitte geben Sie einen Betrag zwischen 1,00 € und Kaufpreis minus 1,00 € ein.“ unter dem Eingabefeld ausgegeben.

User-Story: 2 Zinssatz zuordnen

Als Verkäufer möchte ich sicherstellen, dass die Zinssätze entsprechend der Laufzeit korrekt zugeordnet werden.

Der Zinssatz ist abhängig von der Laufzeit in Monaten.

- 6% für 6-12 Monate
- 3% für 13-24 Monate
- 2% für 25-48 Monate

Die Eingabe von Laufzeit in Monaten erfolgt als ganze Zahl, gültig ist die Eingabe von 6 bis 48 Monaten.

Akzeptanzkriterien:

- 1 Die Eingabe von Laufzeit in Monaten ist nur als ganze Zahl möglich.
- 2 Die Eingabe der Werte von 6 – 48 wird für Laufzeit in Monaten akzeptiert.
- 3 Nach Eingabe von Laufzeit in Monaten wird der Zinssatz zugeordnet.
- 4 Bei Eingabe eines ungültigen Wertes für Laufzeit in Monaten wird die Fehlermeldung „Bitte geben Sie eine Laufzeit zwischen 6 und 48 Monaten ein.“ unter dem Eingabefeld ausgegeben.

siehe Übungsheft,
Seite 61-65

Akzeptanzkriterien

Akzeptanzkriterien

- sind die Bedingungen, die eine Implementierung der User-Story erfüllen muss, um von den Stakeholdern akzeptiert zu werden.
- können als die Testbedingungen betrachtet werden, die durch die Tests geprüft werden sollten.
- sind in der Regel ein Ergebnis der Diskussion (siehe 3 C's).



Akzeptanzkriterien

Akzeptanzkriterien werden verwendet, um ...

- den Umfang der User-Story zu definieren.
- einen Konsens zwischen den Stakeholdern zu erreichen.
- sowohl positive als auch negative Szenarien zu beschreiben.
- als Basis für Abnahmetests der User-Story zu dienen.
- eine genaue Planung und Schätzung zu ermöglichen.



Formulieren von Akzeptanzkriterien

Es gibt mehrere Möglichkeiten, Akzeptanzkriterien für eine User-Story zu formulieren.
Die zwei gängigsten Formate sind:

Szenario-orientiert

(z. B. das Gegeben/Wenn/Dann-Format, das in der verhaltensgetriebenen Entwicklung (BDD) verwendet wird)

Regelorientiert

(z. B. Verifizierungsliste mit Aufzählungspunkten oder tabellarische Form der Input-Output-Zuordnung)

Die meisten Akzeptanzkriterien lassen sich in einem dieser beiden Formate dokumentieren.
Das Team kann andere, benutzerdefinierte Formate verwenden, solange die Akzeptanzkriterien klar definiert und eindeutig sind.

Szenario-orientierte Akzeptanzkriterien



User-Story 4: Darlehensbetrag, Zinssatz und Ratenhöhe anzeigen

Als Kunde möchte ich mir die monatliche Ratenhöhe der Autofinanzierung anzeigen lassen, um zu beurteilen, ob ich mir die Autofinanzierung leisten kann.

Bei Eingabe von Anzahlung und Laufzeit in Monaten wird der Darlehensbetrag, der Zinssatz und die Ratenhöhe ermittelt und angezeigt.

Akzeptanzkriterien:

language: de

Funktionalität: Virtual Showroom II - Finanzierung

Vorbedingungen:

Gegeben seien ein Kunde ist in "VIRTUAL SHOWROOM II" eingeloggt
Und ein Fahrzeug ist konfiguriert #Testfahrzeug 1= 21.600 Euro
Und der Dialog „VSR-II EASY Finance - Ratenkredit“ ist geöffnet

Szenarien 1: Finanzierung ermitteln

Wenn Eingabe von <Anzahlung>
Und Eingabe von <Laufzeit in Monaten>

Dann wird der <Darlehensbetrag> ermittelt und angezeigt
Und die <monatliche Ratenhöhe> wird ermittelt und angezeigt
Und der <Zinssatz> wird ermittelt und angezeigt

Beispiele:

Anzahlung	Laufzeit in Monaten	Darlehensbetrag	mtl. Ratenhöhe	Zinssatz
1	10	21.599	2.289,49	6
21.599	20	1	0,05	3

siehe Übungsheft,
Seite 65

Szenario-orientierte Akzeptanzkriterien



1: Describe behaviour in plain text

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers
```

```
Scenario: Add two numbers:
  Given I have entered 50 into the calculator
  And I have entered 70 into the calculator
  When I press add
  Then the result should be 120 on the screen
```

2: Write a step definition in Ruby

```
Given I have entered {{n}} into the calculator do |n|
  calculator = Calculator.new
  calculator.push(n.to_i)
end
```

3: Run and watch it fail

```
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers
Scenario: Add two numbers # features/addition.feature
Given I have entered 50 into the calculator # features/step_definitions/calculator_steps.rb:2-in `Given /
features/addition.feature:7-in 'Given I have entered 50 into
And I have entered 70 into the calculator' # features/step_definitions/calculator_steps.rb:2-in `And /
When I press add' # features/step_definitions/calculator_steps.rb:2-in `When /
Then the result should be 120 on the screen' # features/step_definitions/calculator_steps.rb:2-in `Then /
```

4. Write code to make the step pass

```
class Calculator
  def push(n)
    @args += [n]
    @args <= n
  end
end
```

5. Run again and see the step pass

```
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers
Scenario: Add two numbers # features/addition.feature
Given I have entered 50 into the calculator # features/step_definitions/calculator_steps.rb:2-in `Given /
features/addition.feature:7-in 'Given I have entered 50 into
And I have entered 70 into the calculator' # features/step_definitions/calculator_steps.rb:2-in `And /
When I press add' # features/step_definitions/calculator_steps.rb:2-in `When /
Then the result should be 120 on the screen' # features/step_definitions/calculator_steps.rb:2-in `Then /
```

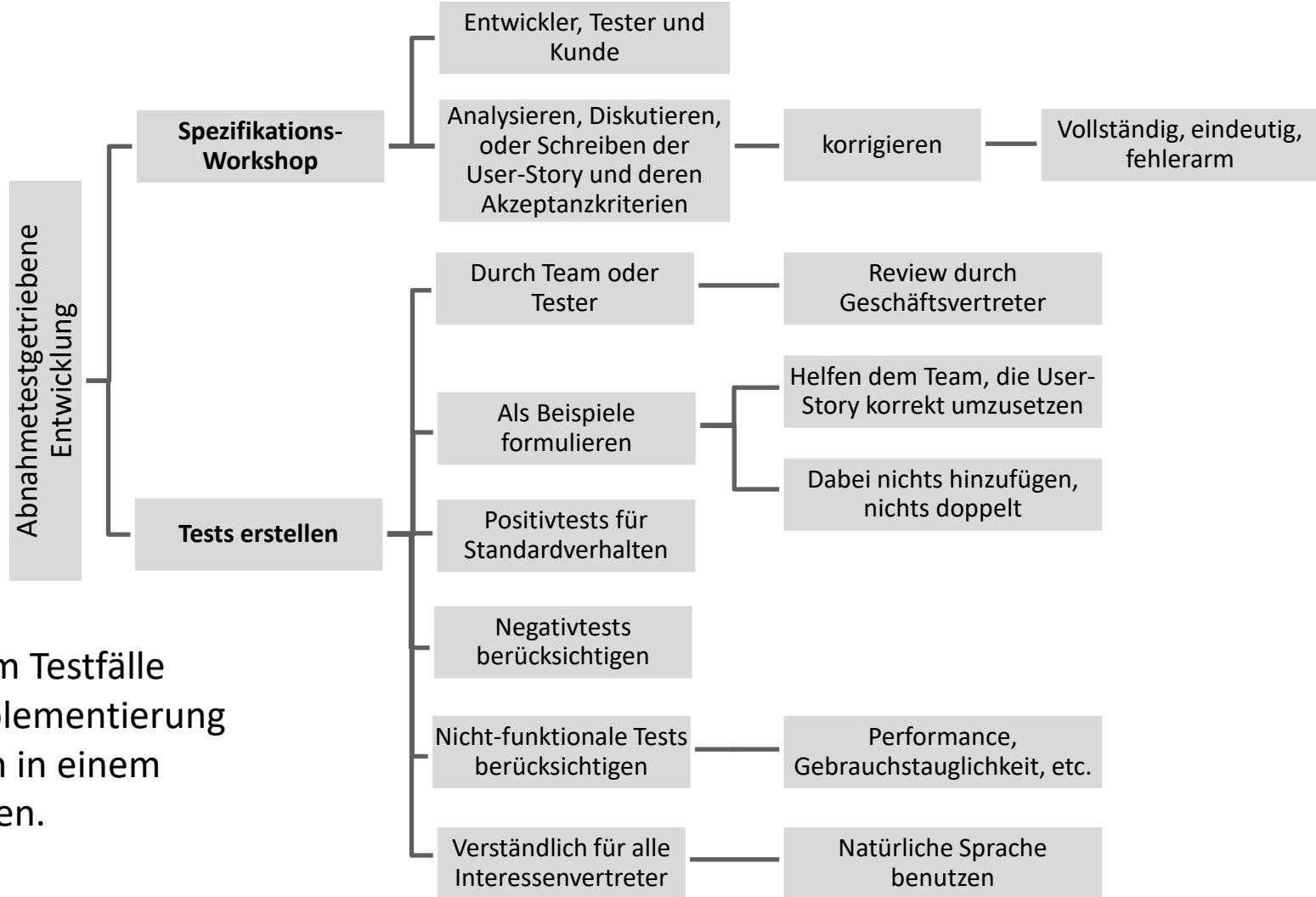
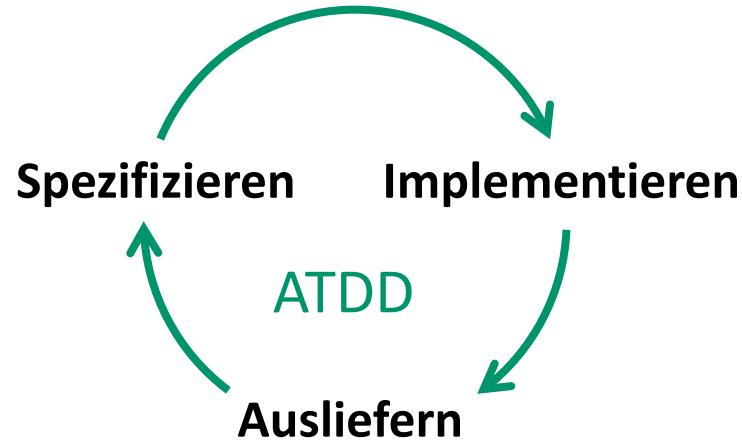
6. Repeat 2-5 until green like a cuke

```
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers
Scenario: Add two numbers # features/addition.feature
Given I have entered 50 into the calculator # features/step_definitions/calculator_steps.rb:2-in `Given /
features/addition.feature:7-in 'Given I have entered 50 into
And I have entered 70 into the calculator' # features/step_definitions/calculator_steps.rb:2-in `And /
When I press add' # features/step_definitions/calculator_steps.rb:2-in `When /
Then the result should be 120 on the screen' # features/step_definitions/calculator_steps.rb:2-in `Then /
```



Abnahmetestgetriebene Entwicklung (ATDD)

Abnahmetestgetriebene Entwicklung (ATDD)



ATDD ist ein Test-First-Ansatz, bei dem Testfälle (manuell oder automatisiert) vor Implementierung der User-Story vom Team gemeinsam in einem Spezifikationsworkshop erstellt werden.



ATDD für den VSR-II-Finanzierung

Für das Epic „Finanzierung“ werden vom Team folgende Akzeptanzkriterien festgelegt.

Abnahmekriterien (Akzeptanzkriterien):

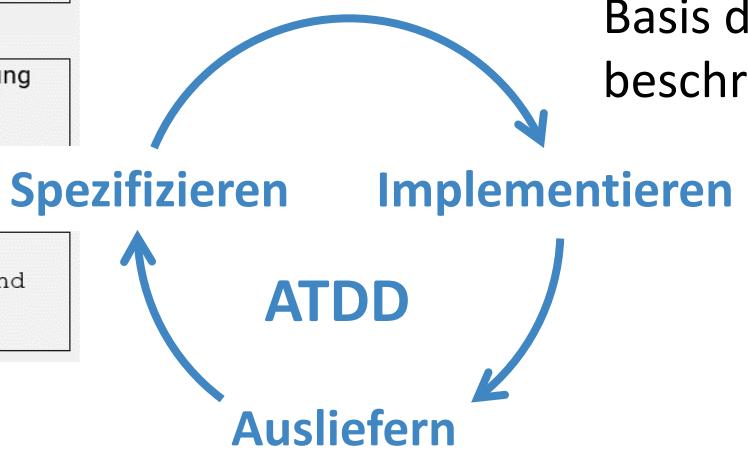
1 Nach Eingabe von Anzahlung und Laufzeit in Monaten wird der Darlehensbetrag, die Ratenhöhe und der Zinssatz ermittelt und angezeigt.

2 Bei Eingabe von ungültigen Werten für Anzahlung wird die Fehlermeldung „Bitte geben Sie einen Betrag zwischen 1,00 € und Kaufpreis minus 1,00 € ein.“ ausgegeben.

3 Bei Eingabe von ungültigen Werten für Laufzeit in Monaten wird die Fehlermeldung „Bitte geben Sie eine Laufzeit zwischen 6 und 48 Monaten ein.“ ausgegeben.

Vor der Implementierung werden vom Team Akzeptanztestfälle erstellt.

Das Team implementiert die Funktion auf Basis der im Epic und in den User-Storys beschriebenen Akzeptanzkriterien.



Im Sprint-Review prüft das Team gemeinsam mit dem Product-Owner, ob die Funktion VSR-II Finanzierung ausgeliefert werden kann.



3-5 Abnahmetestgetriebene Entwicklung (ATDD)



Für das EPIC „Finanzierung“ wurden vom Team die Akzeptanzkriterien diskutiert und dokumentiert.

Anschließend wurden Testfälle auf Basis der Akzeptanzkriterien erstellt.

- Entscheiden Sie, welche Testfälle sinnvoll sind.
- Verwenden Sie das im **Übungsheft** enthaltene Schema – oder öffnen Sie die Übung als **Microlearning** über den QR-Code:



Schlüsselbegriffe – Auf Zusammenarbeit basierende Testansätze



Akzeptanzkriterien

Diejenigen Kriterien, die eine Komponente oder ein System erfüllen muss, um durch den Benutzer, Kunden oder eine bevollmächtigte Instanz abgenommen zu werden.

abnahmetestgetriebene Entwicklung

Ein auf Zusammenarbeit basierender Test-First-Ansatz, der Abnahmetests in der Fachsprache der Stakeholder definiert.

auf Zusammenarbeit basierender Testansatz

Ein Testansatz, der durch Zusammenarbeit zwischen Stakeholdern auf Vermeidung von Fehlerzuständen fokussiert.



User-Storys schreiben

Als [Rolle] möchte ich,
dass [das zu erreichende
Ziel], so dass ich
[resultierender Nutzen für
die Rolle]

Akzeptanzkriterien

- ...
- ...

User-Story = Feature, das für einen Benutzer eines Systems oder einer Software nützlich sein wird

Ziel = Gemeinsame Vision von dem, was geliefert werden soll

→ Durch Berücksichtigung der drei Perspektiven: Fachlichkeit, Entwicklung und Testen

Verfahren zur Erstellung

- Brainstorming
- Mind-Mapping

Drei kritische User-Story-Aspekte

- **Karte (Card)**
das Medium, das eine User-Story beschreibt
- **Konversation (Conversation)**
erklärt, wie die Software genutzt werden soll
- **Bestätigung (Confirmation)**
die Akzeptanzkriterien



User-Story – Akzeptanzkriterien

Akzeptanzkriterien für eine User-Story sind die Bedingungen, die eine Implementierung der User-Story erfüllen muss, um von den Stakeholdern akzeptiert zu werden.

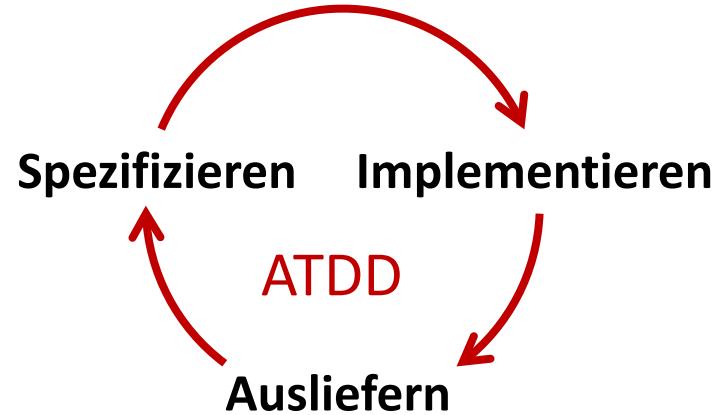
Es gibt mehrere Möglichkeiten, Akzeptanzkriterien für eine User-Story zu formulieren.
Die zwei gängigsten Formate sind:

- **Szenario-orientiert** z. B. das Gegeben/Wenn/Dann-Format, das in der verhaltensgetriebenen Entwicklung (BDD) verwendet wird
- **Regelorientiert** z. B. Verifizierungsliste mit Aufzählungspunkten oder tabellarische Form der Input-Output-Zuordnung

Es können auch andere, benutzerdefinierte Formate verwendet werden,
solange die Akzeptanzkriterien klar definiert und eindeutig sind (z.B. Keywords).



Abnahmetestgetriebene Entwicklung (ATDD)



- ATDD ist ein Test-First-Ansatz
- Testfälle werden **vor** der Implementierung der User-Story erstellt
- Testfälle werden von Teammitgliedern mit unterschiedlichen Perspektiven erstellt, z. B. von Kunden, Entwicklern und Testern
- Testfälle können manuell oder automatisiert ausgeführt werden

Vorgehen zur Testfallableitung

1. Spezifikationsworkshop

- User-Story und Akzeptanzkriterien werden von den Teammitgliedern analysiert, diskutiert und geschrieben
- Unvollständigkeiten, Mehrdeutigkeiten oder Fehlerzustände in der User-Story werden behoben

2. Erstellung der Testfälle

- Durch Team oder einen einzelnen Tester
- Testfälle basieren auf den Akzeptanzkriterien
→ Dienen als Beispiele für die Funktionsweise der Software