
Vorlesung „Methoden des Software Engineering“

Block D „Qualitätssicherung“

Black-Box-Test und White-Box-Test

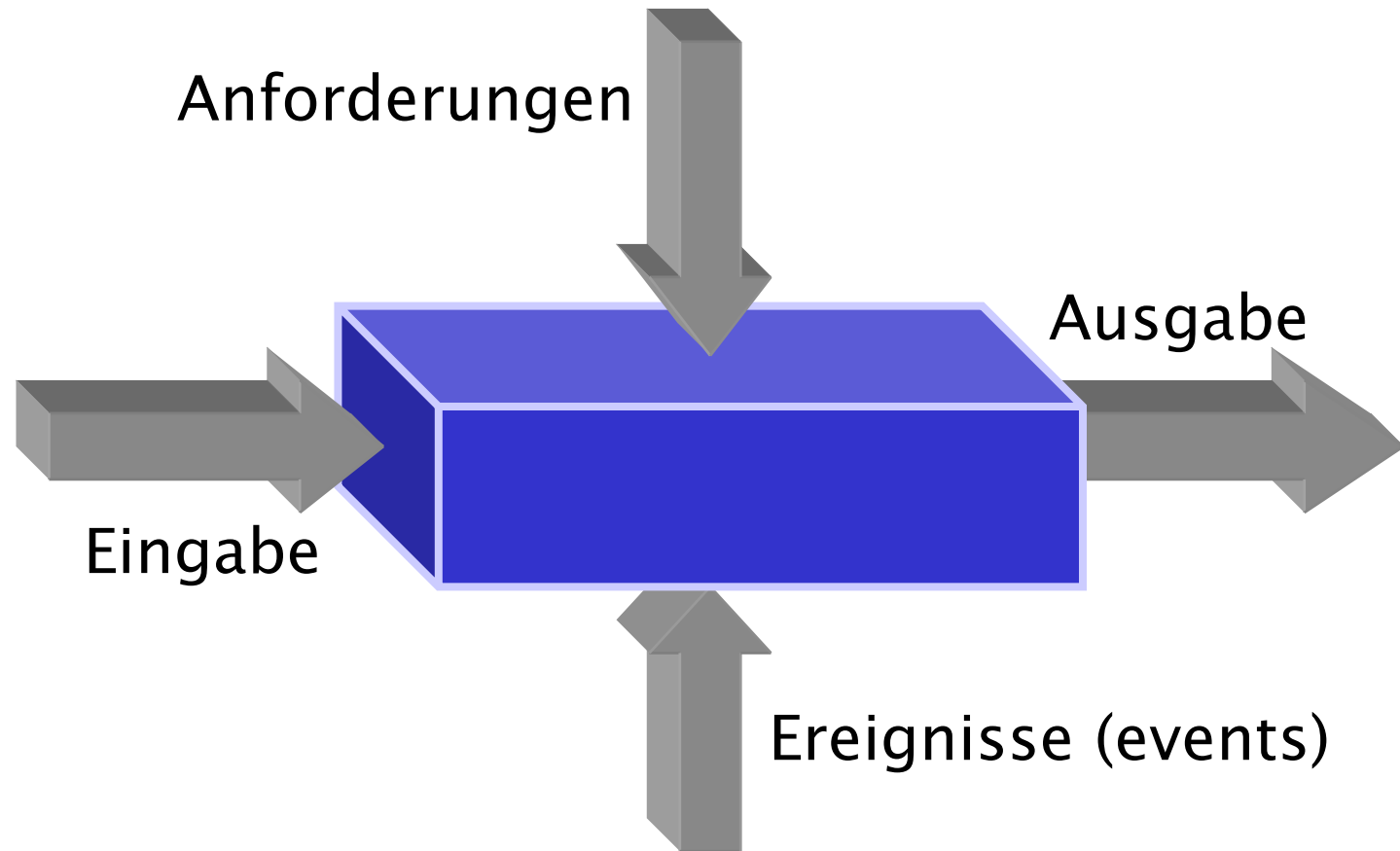
Martin Wirsing

Einheit D.2, 11.01.2007

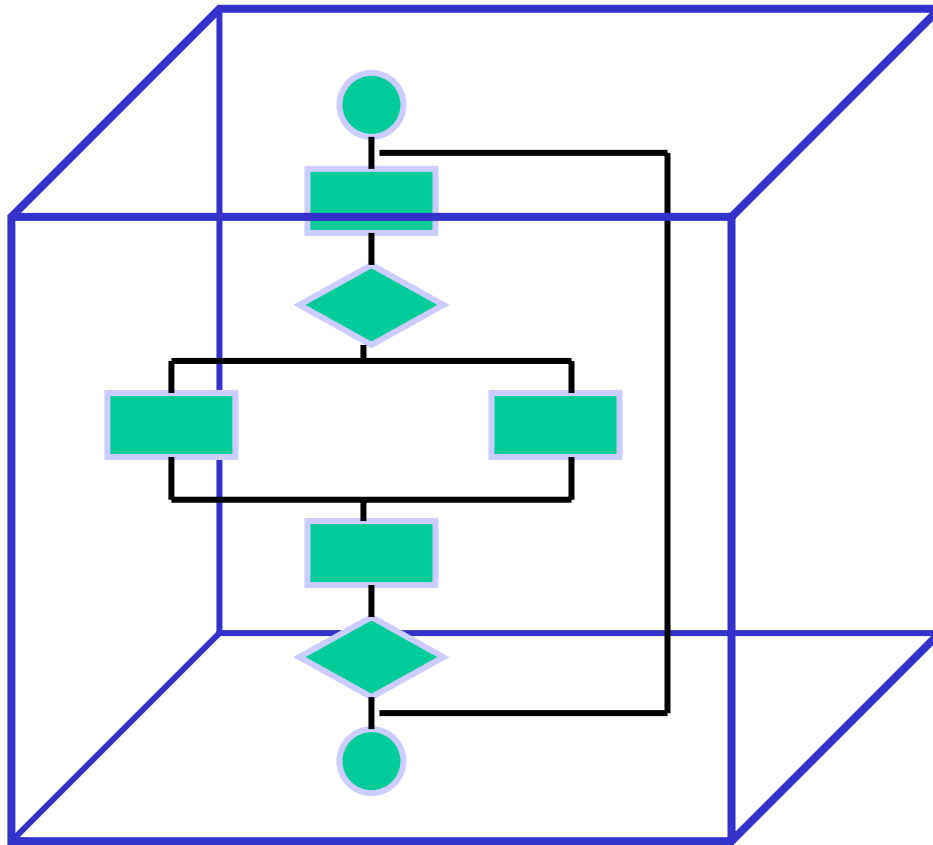
Ziele

- Techniken des Black-Box-Testens und des White-Box-Testens kennen lernen

Black-Box-Test



White-Box-Test



... Ziel ist, dass alle Anweisungen, Bedingungen und Pfade mindestens einmal ausgeführt wurden...

Testen

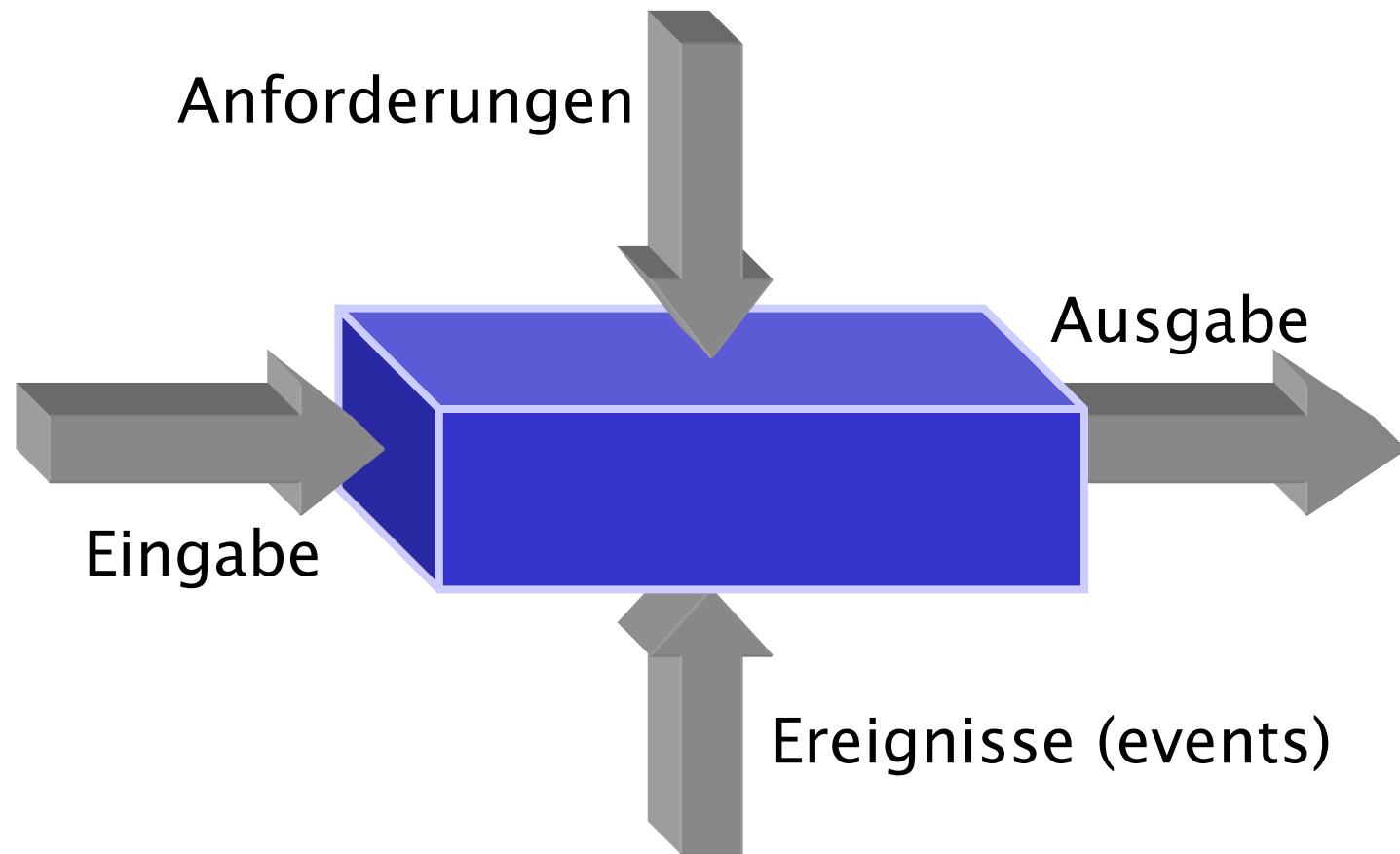
Black-Box-Test	White-Box-Test
<ul style="list-style-type: none">• Testfälle gehen von der Spezifikation aus• Interna des Testobjekts sind bei der Ermittlung der Testfälle unbekannt• Testüberdeckung wird an Hand des spezifizierten Ein/Ausgabeverhaltens gemessen	<ul style="list-style-type: none">• Testfälle ausgehend von der Struktur des Testobjekt• Testfälle werden vom Entwickler beschrieben• Testüberdeckung wird an Hand des Codes gemessen

Testen

- **Methoden des Black-Box-Test**
 - Äquivalenzklassenmethode
 - Methode der Grenzwertanalyse
 - Test von Zustandsautomaten (*wird später behandelt*)

- **Methoden des White-Box-Test**
 - kontrollflussorientierte Verfahren
 - Anweisungsüberdeckungsverfahren
 - Kantenüberdeckungsverfahren
 - Bedingungsüberdeckungsverfahren
 - Pfadüberdeckungsverfahren
 - datenflussorientierte Verfahren (*wird nicht behandelt*)

Black-Box-Test



Fehlerklassen beim Black-Box-Testen

- Inkorrekte oder fehlende Funktionen
- Schnittstellenfehler
- Fehler in Datenstrukturen oder externem Datenbankzugriff
- Performanzfehler
- Fehler bei Initialisierung und Terminierung von Abläufen

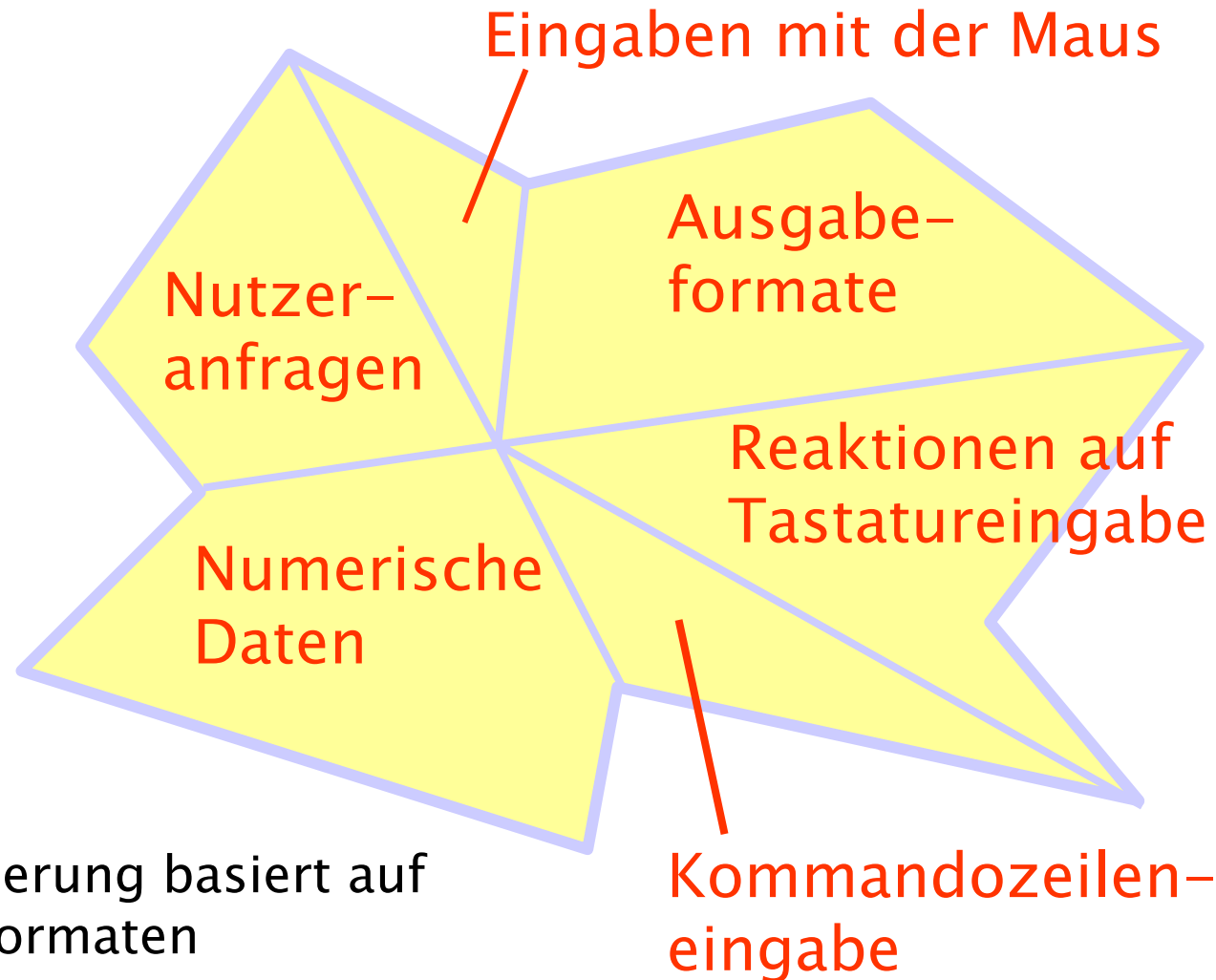
Beispiel

```
int abs (int x)
```

```
{  
    if (x >= 0) return x;  
    if (x < 0)  return x;  
}
```

- Fehlverhalten für alle $x < 0$. Der konkrete Wert von x ist nicht wichtig!
-  Basis für Äquivalenzklassenbildung

Äquivalenzklassentest: Äquivalenzpartitionierung



Partitionierung basiert auf Eingabeformaten

Äquivalenzklassen

- **Gültige Daten**
 - Kommandos der Benutzer
 - Antworten auf Systemausgaben
 - Dateinamen
 - Datenstrukturen des Programms
 - physikalische Parameter
 - Grenzwerte
 - Initialisierungswerte
 - initiation values
 - Kommandos zur Formatierung von Ausgabedaten
 - Antworten auf Fehlermeldungen
 - Graphische Daten (z.B. Mausklicks)
- **Ungültige Daten**
 - Daten außerhalb der Programmgrenzen
 - Physikalisch unmögliche Daten
 - Richtige Werte, die an falscher Stelle eingegeben werden

Äquivalenzklassenmethode und Grenzwertanalyse

- Äquivalenzklassenmethode (heuristisches Verfahren)
 - Eine Äquivalenzklasse ist eine Menge von Eingabewerten, die auf ein Programm eine gleichartige Wirkung ausüben.
 - Es werden Äquivalenzklassen gültiger und ungültiger Werte gebildet, welche den Eingabebereich abdecken.
 - die Testfälle entsprechen (Kombinationen von) Äquivalenzklassen
 - aus jeder Äquivalenzklasse wird mindestens ein Testdatum gewählt
- Grenzwertanalyse
 - basierend auf der Äquivalenzklassenmethode
 - aus jeder Äquivalenzklasse werden Testdaten gewählt, welche Grenzwerte der Äquivalenzklassen abdecken

Äquivalenzklassenmethode: Beispiel 1

Funktion zur Bestimmung der Anzahl der Tage in einem Monat
Eingabe: int monat, int jahr

- **monat: Äquivalenzklassen gültiger Werte**

$C_{m,31}$ = Monate mit 31 Tagen = {1,3,5,7,8,10,12}

$C_{m,30}$ = Monate mit 30 Tagen = {4,6,9,11}

$C_{m,feb}$ = {2}

Ungültige Werte: Negative Zahlen, Zahlen > 12

- **jahr: Äquivalenzklassen gültiger Werte**

$C_{j,schaltjahr}$ = Menge der Schaltjahre

$C_{j,nicht-schaltjahr}$ = Menge der Nicht-Schaltjahre

Ungültige Werte: Negative Zahlen

Äquivalenzklassenmethode: Beispiel 1

Ableitung von Testfällen gültiger Eingaben aus den Äquivalenzklassen

Testfall	Äquivalenzklasse	Ausgabe	ausgewähltes Testdatum		
			monat	jahr	Ausgabe: Soll
T _{1,1}	C _{m,31} und C _{j,nicht-schaltjahr}	31	7	1901	31
T _{1,2}	C _{m,31} und C _{j,schaltjahr}	31	7	1904	31
T _{2,1}	C _{m,30} und C _{j,nicht-schaltjahr}	30	6	1901	30
T _{2,2}	C _{m,30} und C _{j,schaltjahr}	30	6	1904	30
T _{3,1}	C _{m,feb} und C _{j,nicht-schaltjahr}	28	2	1901	28
T _{3,2}	C _{m,feb} und C _{j,schaltjahr}	29	2	1904	29

Äquivalenzklassenmethode: Beispiel 2

Spezifikation (noch nicht ganz konsistent) zur Ableitung des technischen Eintrittsalters einer Person in einen Versicherungsvertrag:

Eingabe: vertragsbeginn, geburtsdatum	
Hilfsvariable	
diff_Monat := Monat(vertragsbeginn) – Monat (geburtsdatum)	
diff_Jahr := Jahr (vertragsbeginn) – Jahr (geburtsdatum)	
technisches_Eintrittsalter	Bedingung
Fehler	vertragsbeginn < geburtsdatum
diff_Jahr	vertragsbeginn >= geburtsdatum und -5 <= diff_Monat <= 6
diff_Jahr + 1	vertragsbeginn >= geburtsdatum und diff_Monat >= 6
diff_Jahr - 1	vertragsbeginn >= geburtsdatum und diff_Monat < - 5

Äquivalenzklassenmethode: Beispiel 2

Test-fall			Ausgewähltes Testdatum		
	Äquivalenzklasse	Ausgabe	Geburtsdatum	Vertragsbeginn	Ausgabe : Soll
T1	1 Vertragsbeginn vor Geburtsdatum	Fehler	01.02.2001	01.01.2001	Fehler
T2	2 diff_Monat im Intervall [-5, 6]	diff_Jahr	01.06.1975	01.08.2001	26
T3	3 diff_Monat ≥ 6	diff_Jahr + 1	01.06.1975	01.12.2001	27
T4	4 diff_Monat < -5	diff_Jahr - 1	01.10.1975	01.01.2001	25

Klasse 1 ist eine Klasse ungültiger Werte

Weitere ungültige Klassen:

Tag, Monat, Jahr nicht im gültigen Wertebereich

Äquivalenzklassenmethode: Beispiel 2

Ti-j : Testfall in Äquivalenzklasse i an der Grenze zu Klasse j

Testfall	Eingabe	Ausgabe	Ausgewähltes Testdatum		
			Geburts-datum	Vertrags-beginn	Ausgabe : Soll
T1-2	Vertragsbeginn 1 Tag vor Geburtsdatum	Fehler			
T2-1	Vertragsbeginn = Geburtsdatum	0			
T2-3	diff_Monat = 6	diff_Jahr			
T2-4	diff_Monat = - 5				
T3-2	diff_Monat = 6	diff_Jahr + 1			
T4-2	diff_Monat = - 6	diff_Jahr - 1			

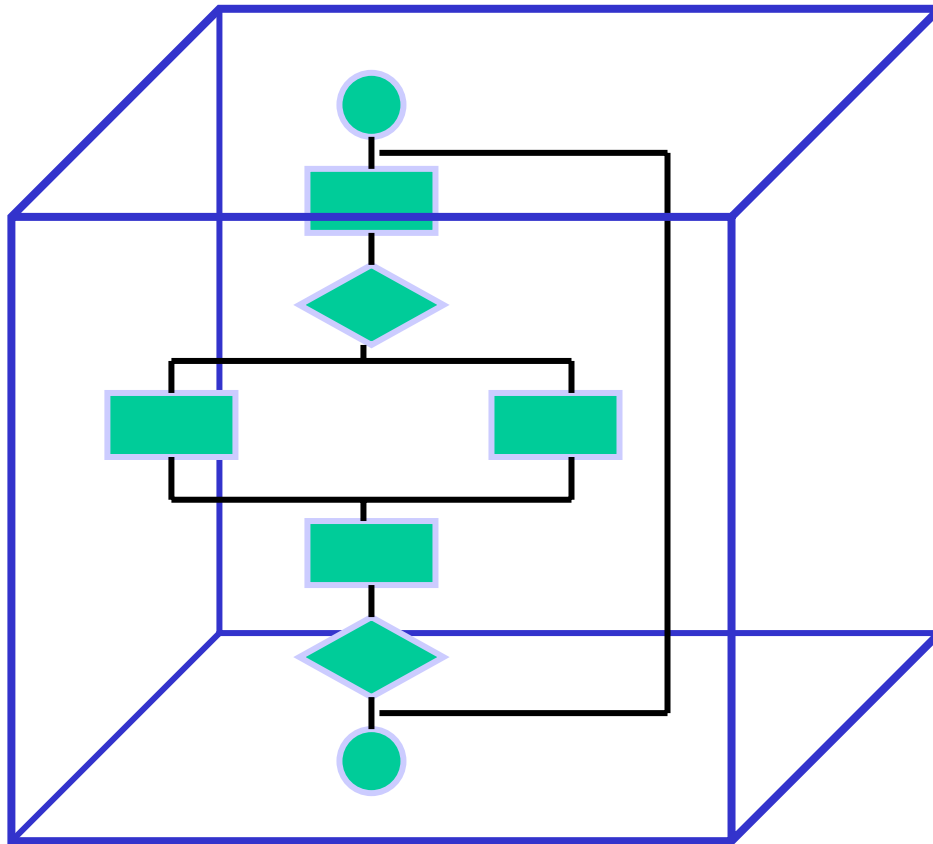
Nebeneffekt:

Es wurde eine Inkonsistenz in der Spezifikation gefunden

Grenzen der Black-Box-Tests

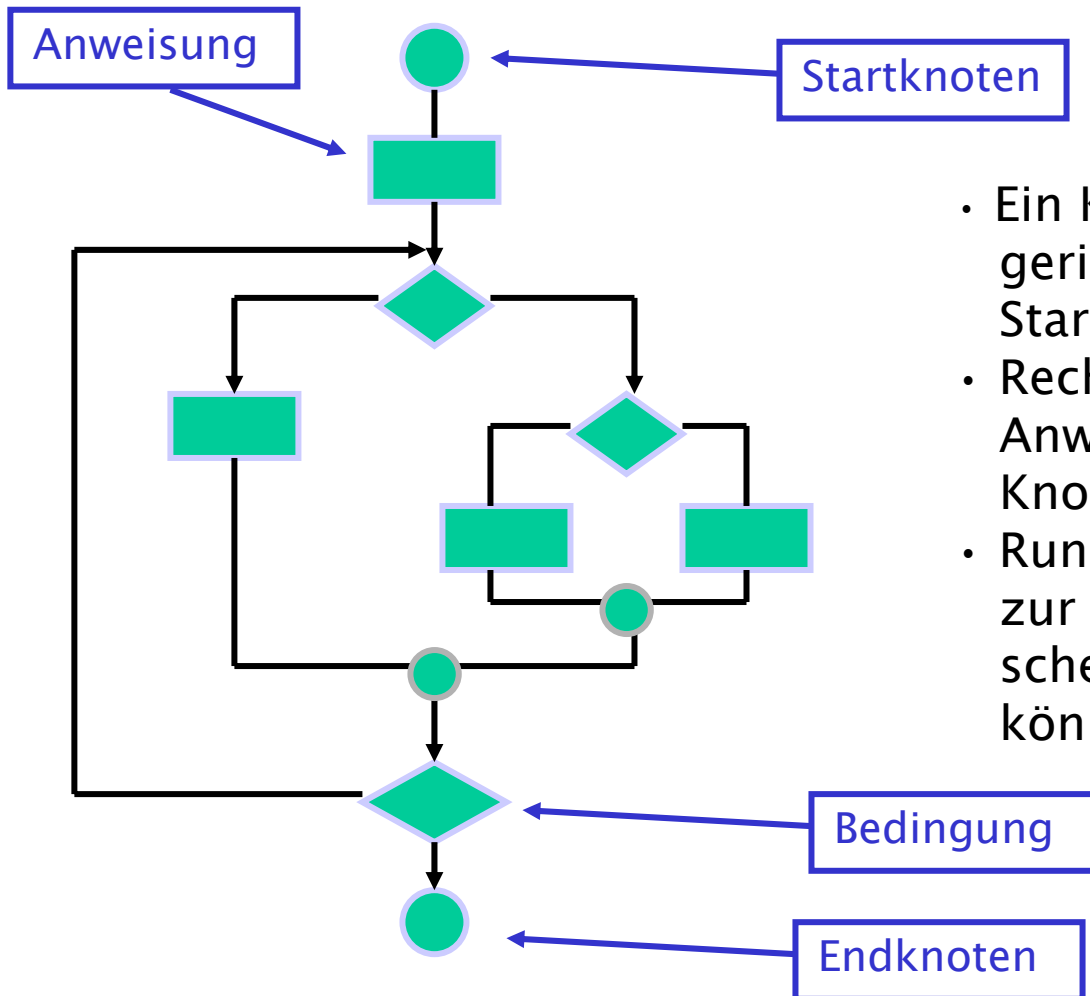
- **Methoden des Black-Box-Test alleine sind nicht ausreichend, da die Spezifikation ein höheres Abstraktionsniveau besitzt wie die Implementierung**
 - nicht alle Elemente einer funktionalen Äquivalenzklasse werden in der Implementierung „intern“ gleich behandelt
 - in der Implementierung kann ein Zustand im Automaten mehreren internen Zuständen entsprechen

White-Box-Test



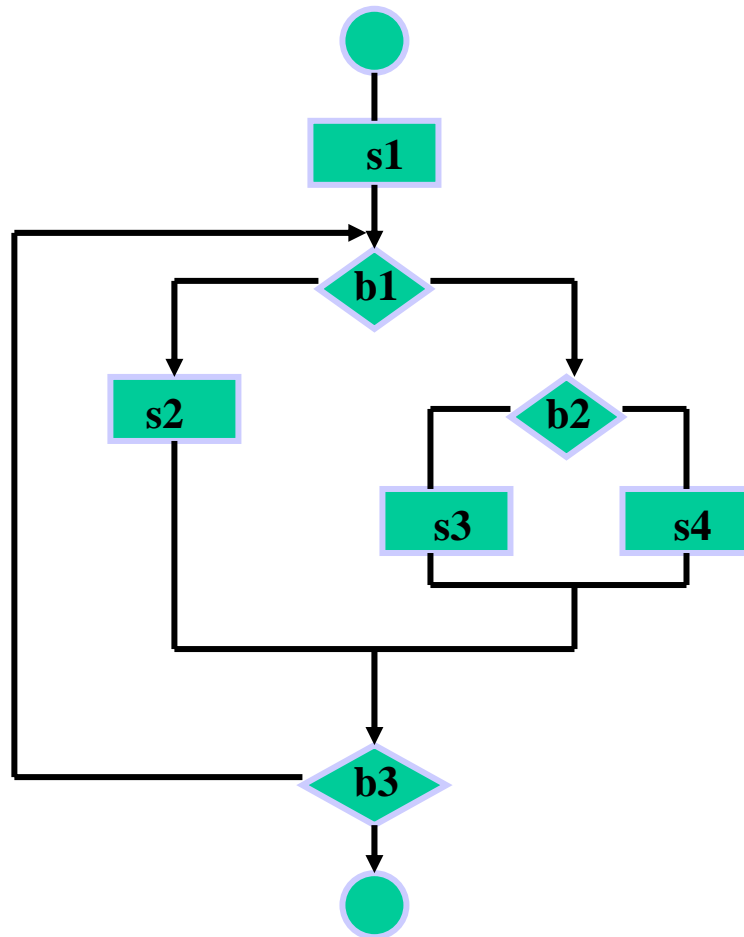
... Ziel ist, dass alle Anweisungen, Bedingungen und Pfade mindestens einmal ausgeführt werden...

Kontrollflussgraph



- Ein Kontrollflussgraph ist ein gerichteter Graph mit genau einem Start- und genau einem Endknoten.
- Rechteckige Knoten stellen Anweisungen dar, rautenförmige Knoten Bedingungen.
- Runde Knoten im Inneren dienen nur zur Zusammenführung von Fallunterscheidungen und können weggelassen werden.

Beispiel Kontrollflussgraph



Kontrollflussgraph

```
s1;  
do {  
  if ( b1 ) s2;  
  else {  
    if ( b2 ) s3;  
    else s4;  
  }  
} while (b3)
```

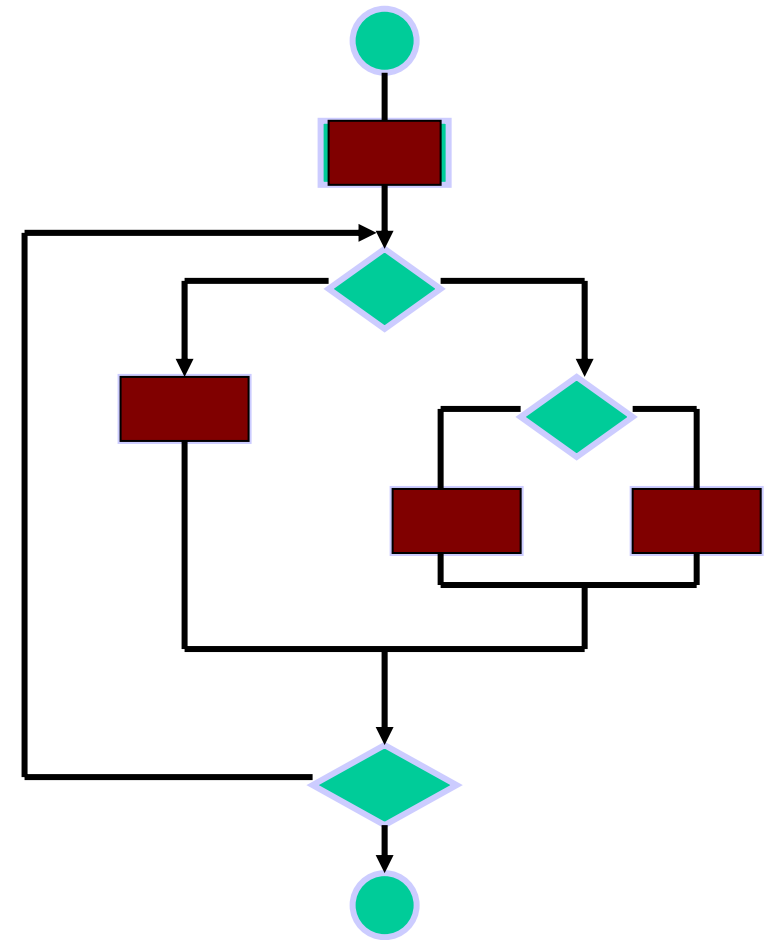
Programmsegment

Kontrollfluss-orientierte Testverfahren

- **Kontrollfluss-orientierte Testverfahren**
orientieren sich am Kontrollflussgraphen des Programms
- Man unterscheidet folgende Typen von Verfahren:
 - Überdeckung von Anweisungen (C0)
 - Kantenüberdeckung (C1)
 - Bedingungsüberdeckung (C2, C3)
 - Pfadüberdeckung (C4)

Anweisungsüberdeckungsverfahren (C0-Test)

- Kriterium: Wähle eine Testmenge, die **alle Anweisungen** des Testobjekts ausführt.
- Es gibt **Werkzeuge** zur Messung der C0-Überdeckung
- **Schwächen** des C0-Verfahrens
 - es müssen nicht alle Wege, die zu einer Anweisung führen überprüft werden: einer genügt
 - unzureichend für den Test von Schleifen



Vollständige C0-Überdeckung

Beispiel eines fehlerhaften Programmfragments (J. Coldewey)

C0- Überdeckung bei Testdaten

T1: ($a < b$ und $a + b < 10$)

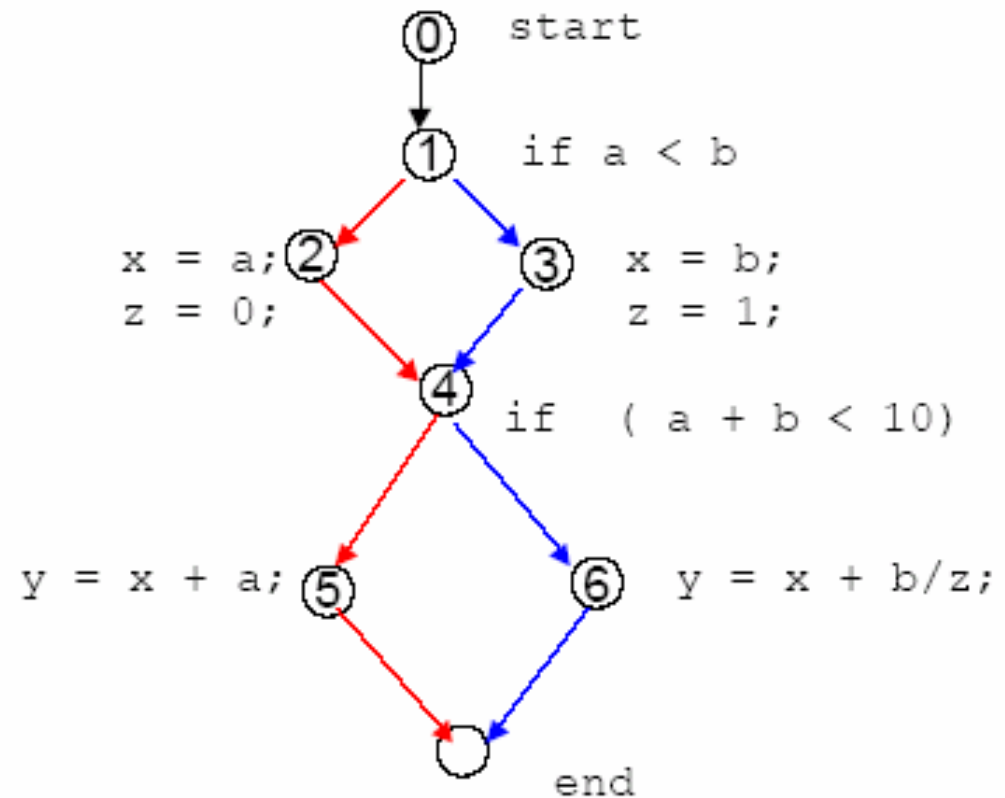
und

T2: ($a \geq b$ und $a + b \geq 10$)

Fehler im Pfad

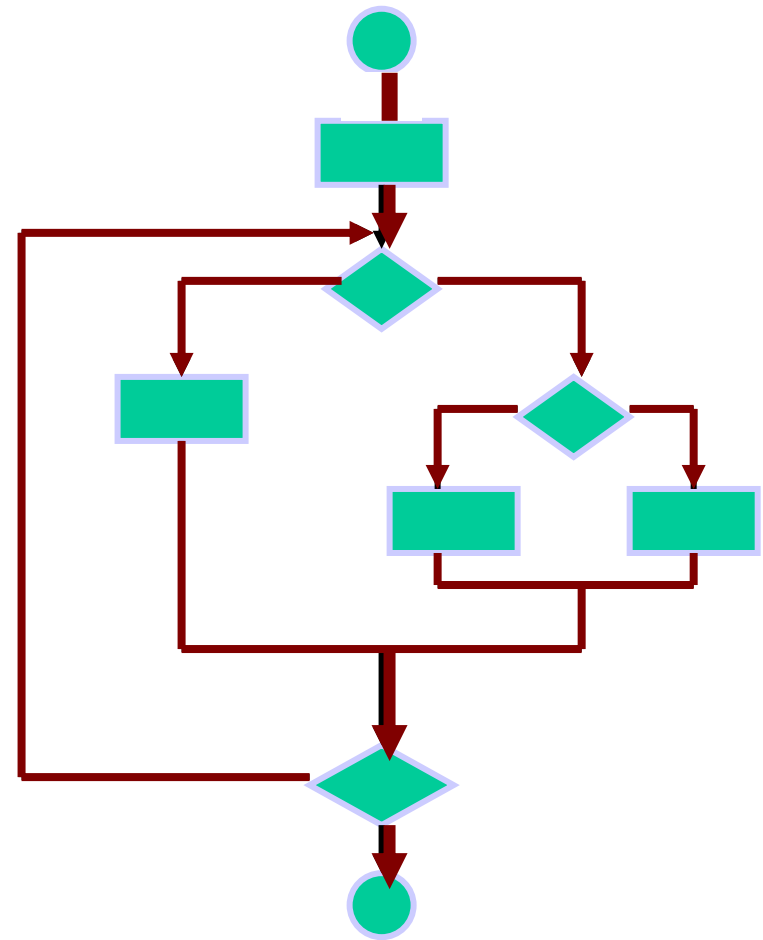
0; 1; 2; 4; 6

wird dabei nicht
entdeckt



Zweigüberdeckungstest (C1-Test)

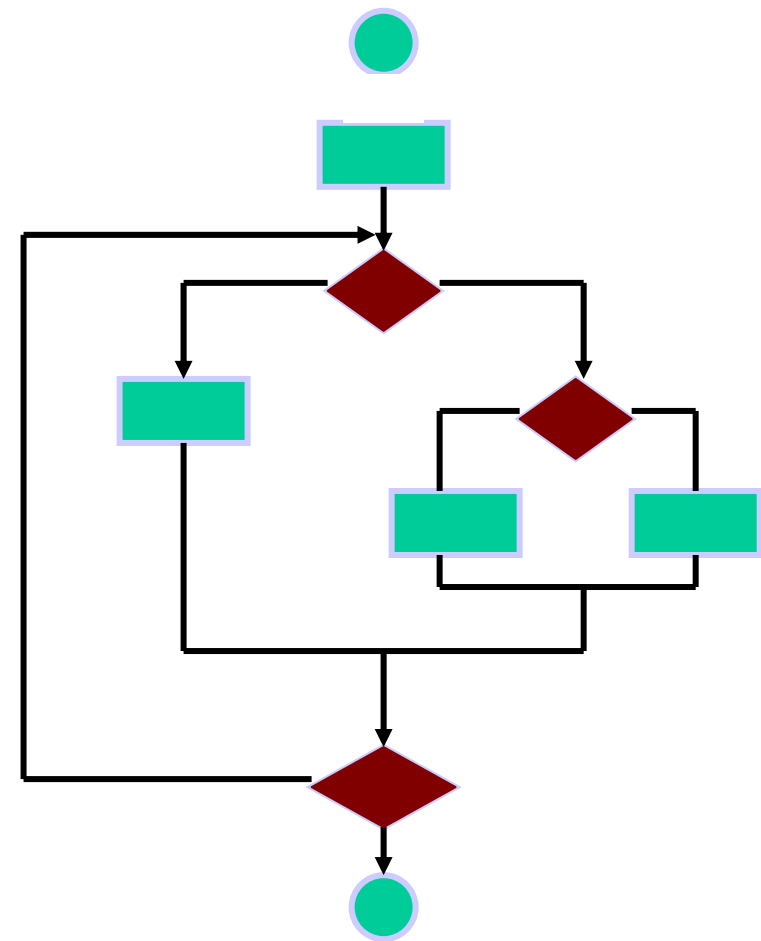
- Kriterium: Wähle eine Testmenge, die **alle Kanten** des Testobjekts ausführt.
- C1-Überdeckung enthält Anweisungsüberdeckungstest und gilt als **Minimalkriterium** für kontrollflussorientiertes Testen (vorgeschrieben vom Luftfahrtstandard RTCA DO-178B ab Stufe B).
- Es gibt **Werkzeuge** zur Messung der C1-Überdeckung
- **Schwächen** des C1-Verfahrens
 - Wie bei C0 müssen nicht alle Wege, die zu einer Anweisung führen überprüft werden;
 - unzureichend für den Test von Schleifen.



Vollständige C1-Überdeckung

Bedingungsüberdeckungstest (C2/C3-Test)

- Kriterium: Wähle eine Testmenge, die **alle Bedingungen** überdeckt.
- Beim **einfachen Test (C2)** werden **alle atomaren Teilformeln** von Bedingungen auf true und false geprüft.
- Beim minimalen **Mehrfach-Bedingungs-Überdeckungstest** werden **zusätzlich alle zusammengesetzten Teilformeln** auf true und false geprüft.
- Beim **Mehrfach-Bedingungs-Überdeckungstest (C3)** werden **alle Wahrheitswertkombinationen atomarer Teilformeln** auf true und false geprüft.



C2/C3-Überdeckung

Beispiel Bedingungsüberdeckung

Bedingung der Form: $(A||B) \&\& C$

Einfacher Bedingungsüberdeckungstest:

	A	B	C	A B	(A B)&&C
1,3	t	-	t	t	t
2,4	t	-	f	t	f
5	f	t	t	t	t
6	f	t	f	t	f
7,8	f	f	-	f	f

- **Sequentielle („unvollständige“) Auswertung:** Die Testfälle 1,3 werden zusammengefasst, da die Belegung von B keine Rolle spielt. Analog für 7,8.
- Die drei Testfälle (1,3), 6 und (7,8) bilden eine **C2-Überdeckung**, da A,B, C jeweils mit `true` und `false` belegt werden.
- Sie bilden auch eine **Überdeckung für den minimalen mehrfachen Bedingungsüberdeckungstest**, da unterschiedliche Werte für A||B vorkommen.

Beispiel Bedingungsüberdeckung

Bedingung der Form: $(A||B) \&\& C$

Modifizierter einfacher Bedingungsüberdeckungstest:

	A	B	C	A B	(A B)&&C
1	t	t	t	t	t
2	t	t	f	t	f
3	t	f	t	t	t
4	t	f	f	t	f
5	f	t	t	t	t
6	f	t	f	t	f
7	f	f	t	f	f
8	f	f	f	f	f

- Der modifizierte C2-Test fordert, dass **jede atomare Teilformel unabhängig von anderen** die Gesamtentscheidung beeinflussen kann.
Bsp. Tests 5 und 7: Die Belegung von B und das Resultat sind verschieden, während die Belegungen für A und C gleich sind
- Dies führt zu (mindestens) **n+1 Testfällen** bei n Teilformeln.
- Dieser Testtyp ist **vorgeschrieben vom Luftfahrtstandard RTCA DO-178B Stufe A**.

Bedingungsüberdeckungstest (C2/C3-Test)

Bemerkungen:

- Bei sequentieller (unvollständiger) Auswertung subsumiert der C2-Test den C1-Test.
- Vollständige Belegung beim C3-Test führt zu exponentiellem Anstieg der Testfälle (im Vergleich zur Anzahl der atomaren Teilformeln einer Bedingung).
- Bedingungsüberdeckungstests sind vor allem interessant, wenn eine komplizierte Verarbeitungslogik vorliegt, die zu kompliziert aufgebauten Entscheidungen führt. Empfohlen wird in diesem Fall der modifizierte C2-Test.

Pfadüberdeckungsverfahren

Kriterium:

Wähle eine Testmenge, die alle Pfade vom Eingangsknoten bis zum Endknoten durchläuft:

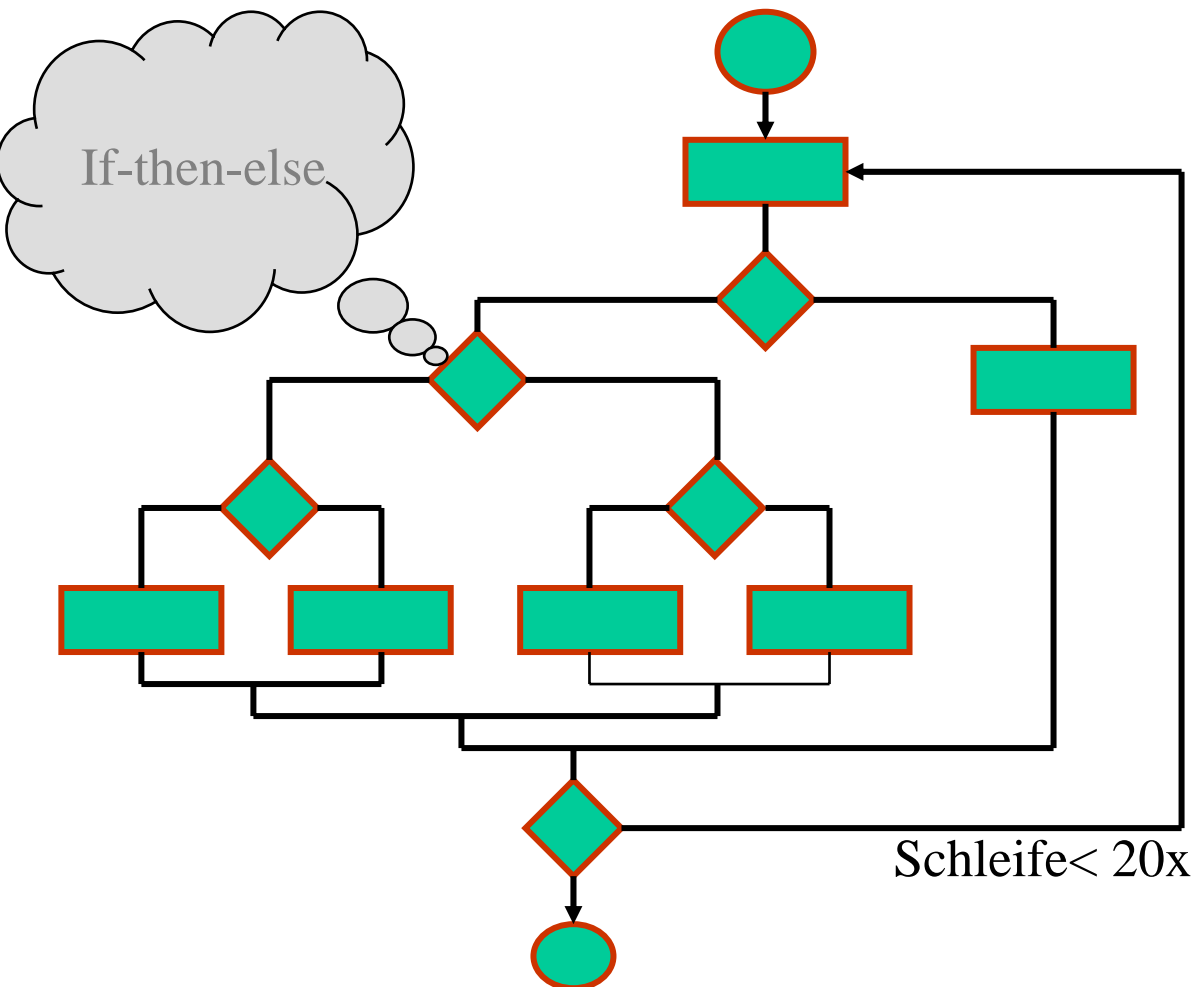
- Nicht realistisch für while-Programme,
- Boundary-interior Pfadtest:

Die Anzahl der Wiederholungen in Schleifen wird eingeschränkt.

Warum alle Pfade überdecken?

- Logische Fehler und inkorrekte Annahmen sind umgekehrt proportional zu der Wahrscheinlichkeit der Ausführung eines Pfads
- Entwickler **glauben** häufig, dass ein bestimmter Pfad nicht ausgeführt wird; die Realität verhält sich häufig anders.
- Tippfehler sind zufällig; sehr wahrscheinlich enthalten ungetestete Pfade Tippfehler.

Garantierte Überdeckung: Alle Pfade



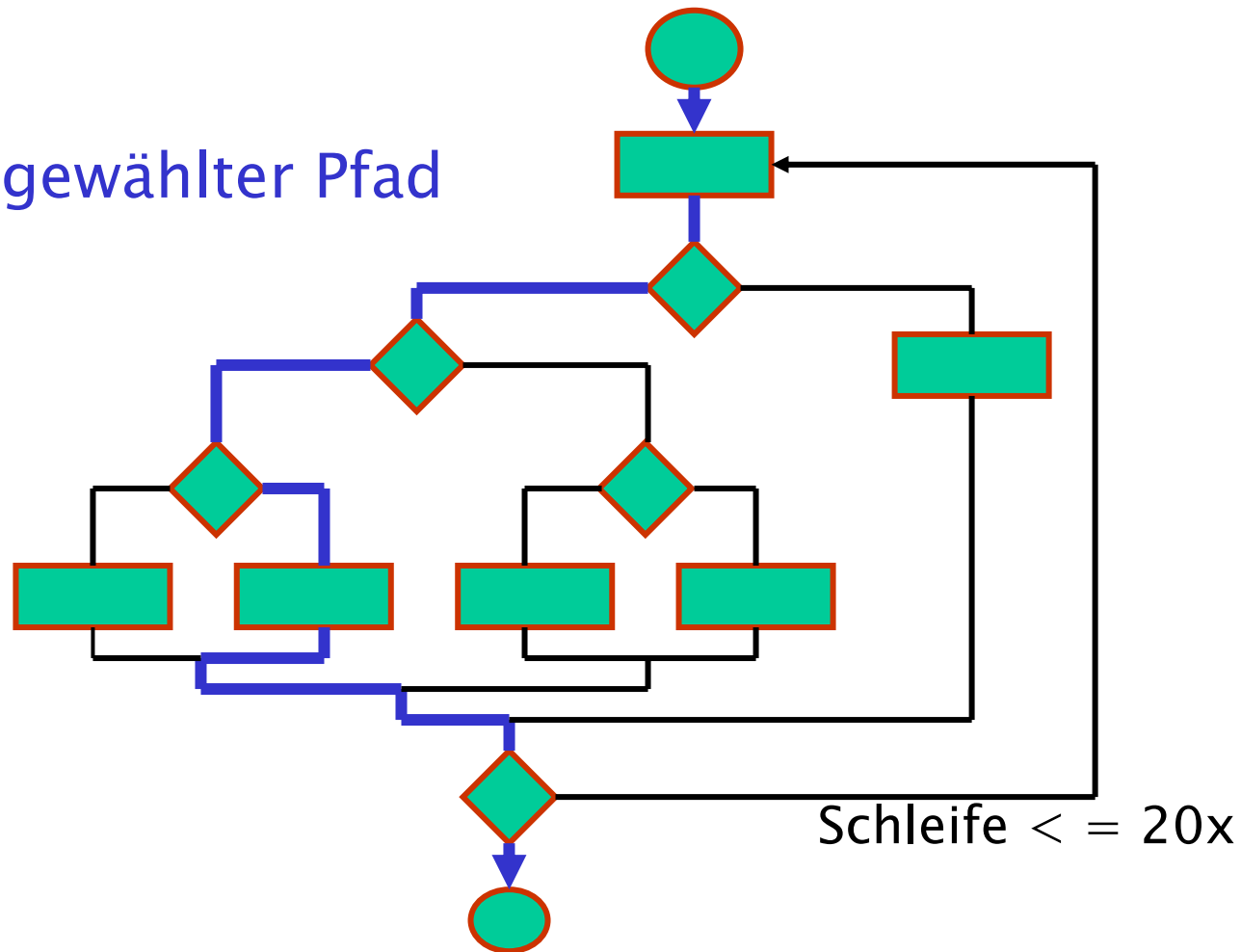
Aber:
Es gibt ca.
 $5^{20} = 10^{14}$
mögliche Pfade!

Wenn möglich, setze
Modellprüfung ein –
z.,B. zusammen mit
Abstraktion.

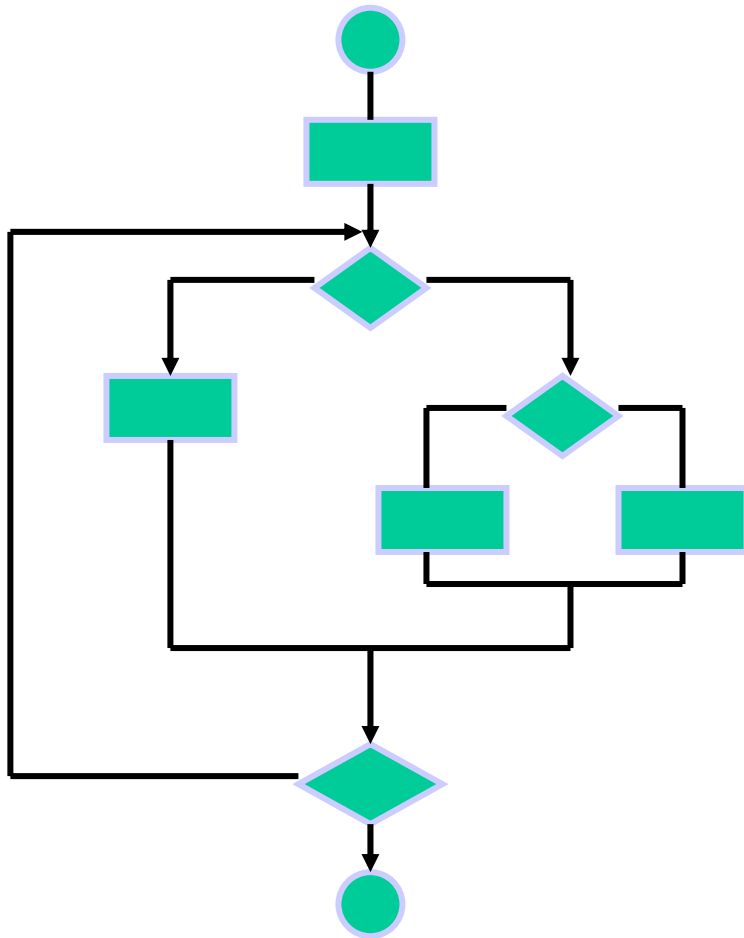
Oder wenn dies **nicht**
erfolgreich:

Selektives Testen

Ein ausgewählter Pfad



Zyklomatische Komplexität: Ein Maß für Pfadüberdeckungstesten



Berechnung der zyklomatischen Komplexität $V(G)$:

Anzahl der Transitionen - Anzahl der Knoten + 2

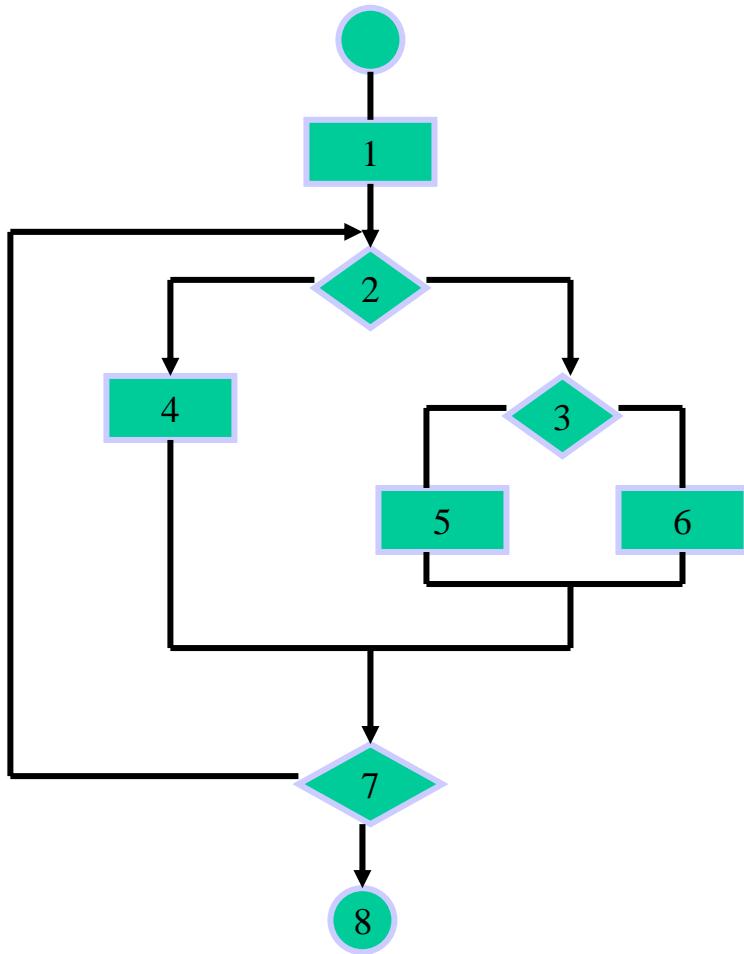
bzw. Anzahl der Bedingungen + 1

In diesem Fall, $V(G) = 4$

$V(G)$ gibt die Anzahl der **linear unabhängigen Zyklen** von G an (wobei Start- und Endknoten miteinander verbunden werden).

$V(G)$ liefert eine **obere Schranke** für die Anzahl der Testfälle, die nötig sind, um alle Anweisungen zu überdecken.

Pfadüberdeckungstesten: Beispiel



$Wg V(G) = 4$ gibt es 4 Pfade:

Pfad 1: 1,2,3,6,7,8

Pfad 2: 1,2,3,5,7,8

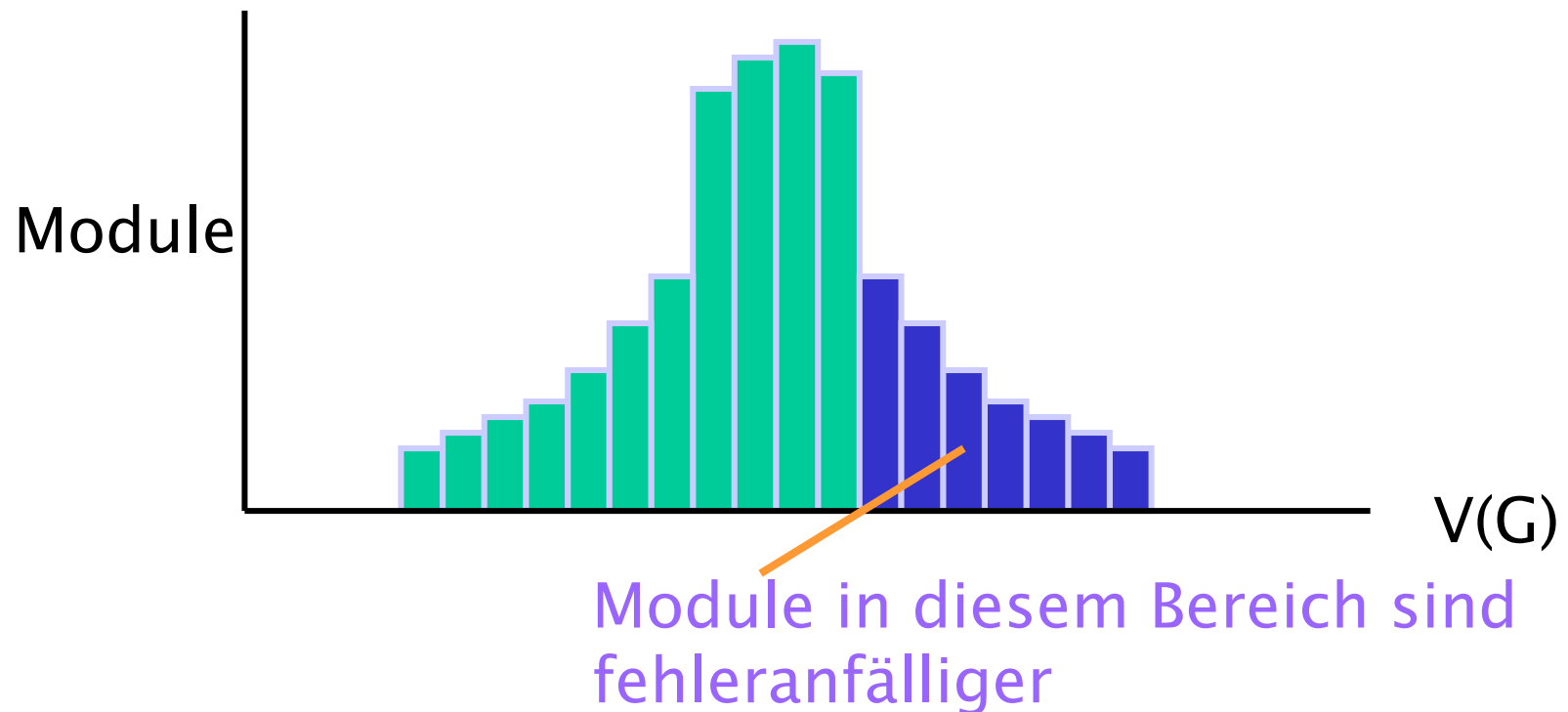
Pfad 3: 1,2,4,7,8

Pfad 4: 1,2,4,7,2,4...7,8

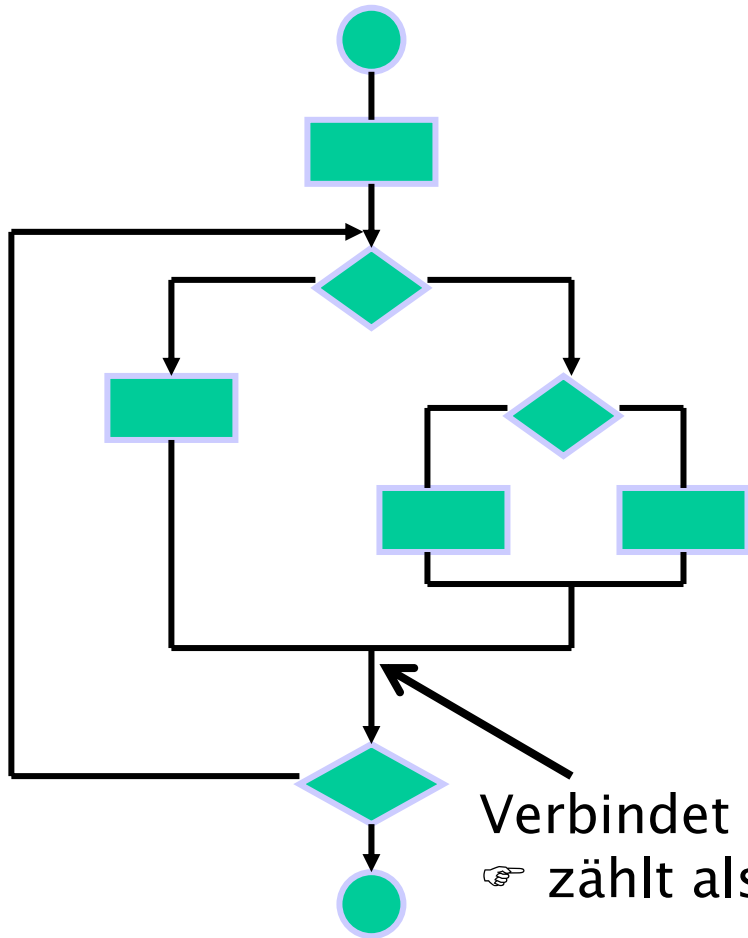
Entwickle daraus Testmengen durch Angabe von geeigneten Inputs.

Zyklomatische Komplexität

Eine Reihe von Industriestudien haben gezeigt, dass die Fehlerwahrscheinlichkeit bei großem $V(G)$ ansteigt.

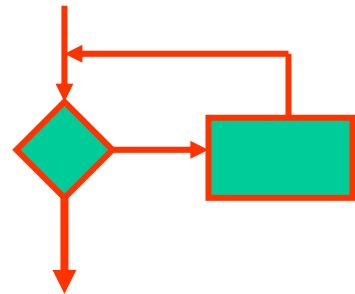


Pfadüberdeckungstesten: Bemerkungen

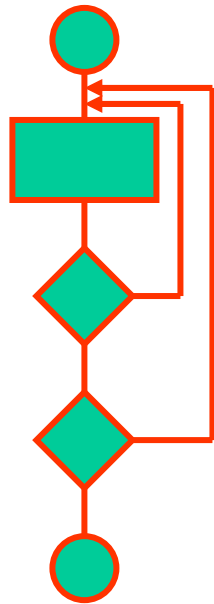


- Pfadüberdeckungstesten sollte bei kritischen Modulen angewendet werden.
- Kontrollflussgraphen sind nicht notwendig, aber hilfreich um die Pfade zu definieren.
- Jede Verbindung zwischen den Kästchen zählt als eine Transition.

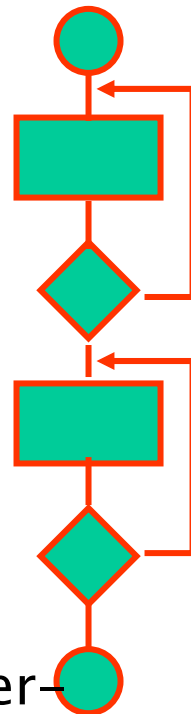
Testen von Schleifen



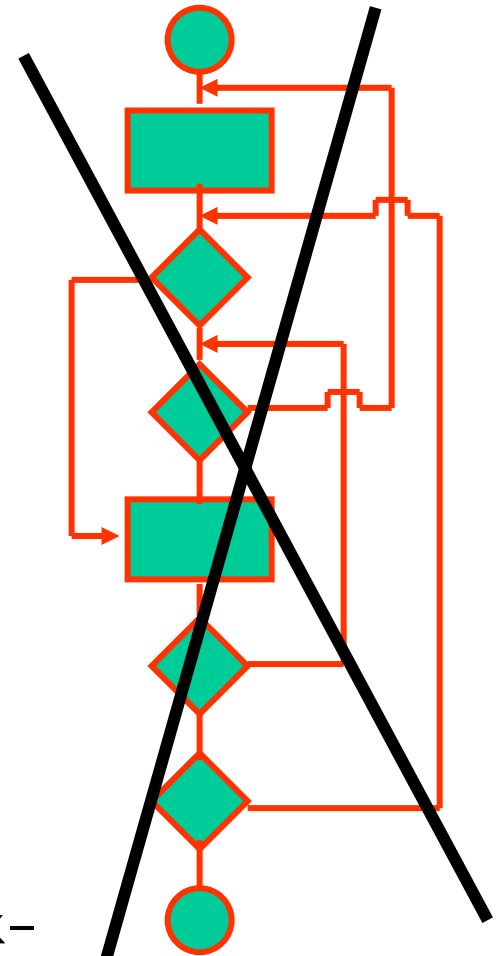
Einfache Schleife



Geschachtelte Schleife



Hintereinanderausführung von Schleifen

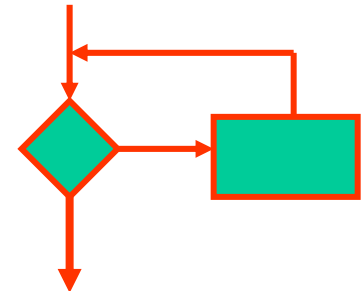


Unstrukturierte Schleife

Testen von Schleifen : Einfache Schleifen

Minimalbedingungen – einfache Schleifen:

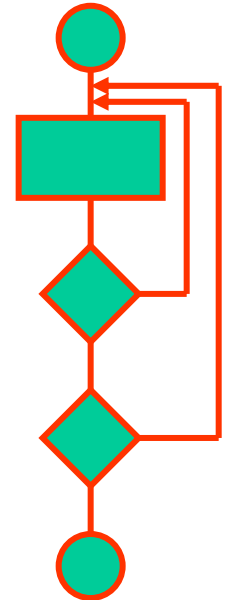
1. KEIN Schleifendurchlauf
2. Nur ein Schleifendurchlauf
3. Zwei Schleifendurchläufe
4. m Schleifendurchläufe mit $m < n$
5. $(n-1)$, n , and $(n+1)$ Schleifendurchläufe



wobei n die maximale Anzahl erlaubter Schleifendurchläufe ist

Geschachtelte Schleifen

- Naïve Erweiterung des Verfahrens für einfache Schleifen:
 - ☞ exponentielle Zunahme der Testfälle
- Reduktion der Testanzahl:
 - Starte mit der innersten Schleife; setze alle anderen Schleifen auf Minimalwerte
 - Führe Tests für einfache Schleife durch mit zusätzlichen Testfällen für Werte außerhalb der Definitionsbereichs und unmögliche Werte
 - Führe Tests für weiter außen liegende Schleifen durch, wobei die inneren Schleifen auf typischen Werten gehalten werden.
 - Fahre so lange fort, bis alle Schleifen getestet sind



Strategien zum White-Box-Testen

- Kontrollfluss-orientierte Tests können nur für kleine Programmsegmente von Hand durchgeführt werden. Sie sind deshalb nur für Unit Tests gut geeignet.
- Metrik-basierte Werkzeuge machen es möglich, White-Box-Tests auch beim Integrations- und Systemtest einzusetzen.

Metrik-gesteuerte Systemtestfall-Auswahlstrategie

- Verwende Äquivalenztests und Grenzwertanalyse für eine vollständige Testsuite von Black-Box-Tests.
- Führe die Black-Box-Testsuite aus und messe den Überdeckungsgrad. Normalerweise wird dieser bei 60% bis 70% Anweisungsüberdeckung liegen.
- Wähle White-Box-Tests so, dass eine bestimmte Überdeckung erreicht wird, wie z.B.
 - >95% Anweisungsüberdeckung und
 - >90% Kantenüberdeckung.

Zusammenfassung

- Das Black-Box-Testen geht von der Spezifikation aus und die Interna des Testobjekts sind nicht bekannt. Das White-Box-Testen ergänzt das Black-Box-Testen durch systematisches Explorieren der Struktur des Testobjekts.
- Wichtige Methoden des Black-Box-Testens sind die Äquivalenzklassenmethode, die Methode der Grenzwertanalyse und der Test von Zustandsautomaten.
- Wichtige Techniken des White-Box-Testens sind kontrollflussorientierte Verfahren wie etwa Anweisungsüberdeckungsverfahren, Kantenüberdeckungsverfahren, Bedingungsüberdeckungsverfahren, Pfadüberdeckungsverfahren sowie datenflussorientierte Verfahren.