



Software-Test: Funktionstest

Andreas Zeller

Lehrstuhl Softwaretechnik
Universität des Saarlandes, Saarbrücken



Funktionale Testverfahren

Funktionale Testverfahren testen gegen die *Spezifikation* und lassen die interne Programmstruktur unberücksichtigt.

- Komplementär zu *white-box*-Verfahren
- Benötigen vollständige und widerspruchsfreie Spezifikation
- Problem: Testvollständigkeit schwierig zu messen



Funktionale Äquivalenzklassenbildung —



2/23

Wertebereiche von Ein- und Ausgaben werden in Äquivalenzklassen eingeteilt

Bildung der Äquivalenzklassen *nur an der Spezifikation* orientiert („black box“)

Äquivalenzklassen für gültige und ungültige Werte

Getestet wird nur noch für jeweils einen Repräsentanten der Klasse



Regeln zur Äquivalenzklassenbildung



- Jeder spezifizierte Eingabebereich induziert mindestens eine gültige und mindestens eine ungültige Klasse

Beispiel: „ i muß kleiner oder gleich 10 sein.“

Gültige Klasse: $i \leq 10$

Ungültige Klasse: $i > 10$.

- Jede Eingabebedingung der Spezifikation induziert eine gültige Klasse (= Bedingung erfüllt) und eine ungültige Klasse (= Bedingung nicht erfüllt)

Beispiel: „Das erste Zeichen muß ein Buchstabe sein.“

Gültige Klasse: Das erste Zeichen ist ein Buchstabe

ungültige Klasse: Das erste Zeichen ist kein Buchstabe



Regeln zur Äquivalenzklassenbildung (2) –

- Bildet eine Eingabebedingung eine Menge von Werten, die unterschiedlich behandelt werden, ist für jeden Fall eine eigene gültige Äquivalenzklasse zu bilden – und eine ungültige.
- Für Ausgaben werden analog Äquivalenzklassen gebildet
- Zum Testen wird *irgendein* Element der Äquivalenzklasse gewählt (am besten zufällig)



Beispiel: ZaehleZchn



Wir betrachten die Spezifikation von ZaehleZchn:

```
void ZaehleZchn(int& VokalAnzahl, int& Gesamtzahl);  
// Lese Zeichen von der Tastatur, bis ein Zeichen  
// gelesen wird, das kein Großbuchstabe ist, oder  
// Gesamtzahl INT_MAX erreicht.  
// Ist das eingelesene Zeichen ein Großbuchstabe,  
// wird Gesamtzahl um 1 erhöht;  
// ist der Großbuchstabe ein Vokal,  
// wird auch VokalAnzahl um 1 erhöht.
```





Beispiel: ZaehleZchn (2)

ZaehleZchn verhält sich in den folgenden Äquivalenzklassen unterschiedlich:

Klasse 1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

Klasse 2 $\text{Gesamtzahl} = \text{INT_MAX}$

Klasse 3 $0 \leq \text{VokalAnzahl} < \text{Gesamtzahl} < \text{INT_MAX}$

Für das eingelesene Zeichen Zchn gilt außerdem:

Klasse 4 Zchn ist kein Großbuchstabe
(Zchn < 'A' oder Zchn > 'Z')

Klasse 5 Zchn ist ein Großbuchstabe

Klasse 6 Zchn ist ein großer Vokal ('A', 'E', ..., 'U')



Beispiel: ZaehleZchn (3)



Die Äquivalenzklassen können noch weiter verfeinert werden:

Eingaben	Gültige Äquivalenzklassen
Gesamtzahl	1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$ 2 $\text{Gesamtzahl} = \text{INT_MAX}$
VokaAnzahl	3 $0 \leq \text{VokaAnzahl} < \text{INT_MAX}$
Zchn	4a $\text{Zchn} < \text{'A'}$ 4b $\text{Zchn} > \text{'Z'}$ 5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$ 6a $\text{Zchn} = \text{'A'}$ 6b $\text{Zchn} = \text{'E'}$ 6c $\text{Zchn} = \text{'I'}$ 6d $\text{Zchn} = \text{'O'}$ 6e $\text{Zchn} = \text{'U'}$





Beispiel: ZaehleZchn (4)

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} \leq \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < \text{'A'}$

4b $\text{Zchn} > \text{'Z'}$

5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$

6a $\text{Zchn} = \text{'A'}$

6b $\text{Zchn} = \text{'E'}$

6c $\text{Zchn} = \text{'I'}$

6d $\text{Zchn} = \text{'O'}$

6e $\text{Zchn} = \text{'U'}$

Mit drei Repräsentanten decken wir alle Äquivalenzklassen ab:

Testfall
Getestete Äquivalenzklasse
Gesamtzahl
VokalAnzahl
Zchn
Soll-Ergebnisse: Gesamtzahl
VokalAnzahl





Beispiel: ZaehleZchn (4)

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} < \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < \text{'A'}$

4b $\text{Zchn} > \text{'Z'}$

5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$

6a $\text{Zchn} = \text{'A'}$

6b $\text{Zchn} = \text{'E'}$

6c $\text{Zchn} = \text{'I'}$

6d $\text{Zchn} = \text{'O'}$

6e $\text{Zchn} = \text{'U'}$

Mit drei Repräsentanten decken wir alle Äquivalenzklassen ab:

Testfall	1
Getestete Äquivalenzklasse	1, 3, 4a, 5, 6a-6e
Gesamtzahl	100
VokalAnzahl	50
Zchn	'X', 'A', 'E', 'I', 'O', 'U', '5'
Soll-Ergebnisse:	
Gesamtzahl	106
VokalAnzahl	5





Beispiel: ZaehleZchn (4)

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} < \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < \text{'A'}$

4b $\text{Zchn} > \text{'Z'}$

5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$

6a $\text{Zchn} = \text{'A'}$

6b $\text{Zchn} = \text{'E'}$

6c $\text{Zchn} = \text{'I'}$

6d $\text{Zchn} = \text{'O'}$

6e $\text{Zchn} = \text{'U'}$

Mit drei Repräsentanten decken wir alle Äquivalenzklassen ab:

Testfall	1	2
Getestete Äquivalenzklasse	1, 3, 4a, 5, 6a-6e	1, 3, 4b
Gesamtzahl	100	1
VokalAnzahl	50	1
Zchn	'X', 'A', 'E', 'I', 'O', 'U', '5'	'a'
Soll-Ergebnisse:		
Gesamtzahl	106	1
VokalAnzahl	5	1





Beispiel: ZaehleZchn (4)

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} < \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < \text{'A'}$

4b $\text{Zchn} > \text{'Z'}$

5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$

6a $\text{Zchn} = \text{'A'}$

6b $\text{Zchn} = \text{'E'}$

6c $\text{Zchn} = \text{'I'}$

6d $\text{Zchn} = \text{'O'}$

6e $\text{Zchn} = \text{'U'}$

Mit drei Repräsentanten decken wir alle Äquivalenzklassen ab:

Testfall	1	2	3
Getestete Äquivalenzklasse	1, 3, 4a, 5, 6a-6e	1, 3, 4b	2, 3
Gesamtzahl	100	1	INT_MAX
VokalAnzahl	50	1	50
Zchn	'X', 'A', 'E', 'I', 'O', 'U', '5'	'a'	-
Soll-Ergebnisse:			
Gesamtzahl	106	1	INT_MAX
VokalAnzahl	5	1	50



Grenzwertanalyse



Grundidee: Repräsentanten sollen *am Rand* der Äquivalenzklasse gewählt werden

Beispiel: Gegeben seien drei Äquivalenzklassen

- $1 \leq \text{aktuellerMonat} \leq 12$ (gültig),
- $\text{aktuellerMonat} < 1$ (ungültig),
- $13 \leq \text{aktuellerMonat}$ (ungültig).

Dann sind 0, 1, 12 und 13 geeignete Repräsentanten.

Hintergrund: Grenzbereiche werden besonders häufig fehlerhaft verarbeitet





Beispiel: ZaehleZchn

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} \leq \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < \text{'A'}$

4b $\text{Zchn} > \text{'Z'}$

5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$

6a $\text{Zchn} = \text{'A'}$

6b $\text{Zchn} = \text{'E'}$

6c $\text{Zchn} = \text{'I'}$

6d $\text{Zchn} = \text{'O'}$

6e $\text{Zchn} = \text{'U'}$

Mit vier Repräsentanten decken wir alle Grenzwerte ab
(U = untere Grenze, O = obere Grenze)

Testfall
Getestete Äquivalenzklasse
Gesamtzahl VokalAnzahl Zchn
Soll-Ergebnisse: Gesamtzahl VokalAnzahl





Beispiel: ZaehleZchn

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} \leq \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < \text{'A'}$

4b $\text{Zchn} > \text{'Z'}$

5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$

6a $\text{Zchn} = \text{'A'}$

6b $\text{Zchn} = \text{'E'}$

6c $\text{Zchn} = \text{'I'}$

6d $\text{Zchn} = \text{'O'}$

6e $\text{Zchn} = \text{'U'}$

Mit vier Repräsentanten decken wir alle Grenzwerte ab
(U = untere Grenze, O = obere Grenze)

Testfall	1
Getestete Äquivalenzklasse	1U, 3U, 4aO, 5U, 5O, 6a-6e
Gesamtzahl	0
VokalAnzahl	0
Zchn	'A', 'E', 'I', 'O', 'U', 'Z', '@'
Soll-Ergebnisse:	
Gesamtzahl	6
VokalAnzahl	5





Beispiel: ZaehleZchn

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} \leq \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < 'A'$

4b $\text{Zchn} > 'Z'$

5 $'A' \leq \text{Zchn} \leq 'Z'$

6a $\text{Zchn} = 'A'$

6b $\text{Zchn} = 'E'$

6c $\text{Zchn} = 'I'$

6d $\text{Zchn} = 'O'$

6e $\text{Zchn} = 'U'$

Mit vier Repräsentanten decken wir alle Grenzwerte ab
(U = untere Grenze, O = obere Grenze)

Testfall	1	2
Getestete Äquivalenzklasse	1U, 3U, 4aO, 5U, 5O, 6a-6e	2, 3O
Gesamtzahl	0	INT_MAX
VokalAnzahl	0	INT_MAX
Zchn	'A', 'E', 'I', 'O', 'U', 'Z', '@'	-
Soll-Ergebnisse:		
Gesamtzahl	6	INT_MAX
VokalAnzahl	5	INT_MAX





Beispiel: ZaehleZchn

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} \leq \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < 'A'$

4b $\text{Zchn} > 'Z'$

5 $'A' \leq \text{Zchn} \leq 'Z'$

6a $\text{Zchn} = 'A'$

6b $\text{Zchn} = 'E'$

6c $\text{Zchn} = 'I'$

6d $\text{Zchn} = 'O'$

6e $\text{Zchn} = 'U'$

Mit vier Repräsentanten decken wir alle Grenzwerte ab
(U = untere Grenze, O = obere Grenze)

Testfall	1	2	3
Getestete Äquivalenzklasse	1U, 3U, 4aO, 5U, 5O, 6a-6e	2, 3O	1, 3U
Gesamtzahl	0	INT_MAX	50
VokalAnzahl	0	INT_MAX	0
Zchn	'A', 'E', 'I', 'O', 'U', 'Z', '@'	-	-
Soll-Ergebnisse:			
Gesamtzahl	6	INT_MAX	50
VokalAnzahl	5	INT_MAX	0





Beispiel: ZaehleZchn

1 $0 \leq \text{Gesamtzahl} < \text{INT_MAX}$

2 $\text{Gesamtzahl} = \text{INT_MAX}$

3 $0 \leq \text{VokalAnzahl} \leq \text{Gesamtzahl} < \text{INT_MAX}$

4a $\text{Zchn} < \text{'A'}$

4b $\text{Zchn} > \text{'Z'}$

5 $\text{'A'} \leq \text{Zchn} \leq \text{'Z'}$

6a $\text{Zchn} = \text{'A'}$

6b $\text{Zchn} = \text{'E'}$

6c $\text{Zchn} = \text{'I'}$

6d $\text{Zchn} = \text{'O'}$

6e $\text{Zchn} = \text{'U'}$

Mit vier Repräsentanten decken wir alle Grenzwerte ab
(U = untere Grenze, O = obere Grenze)

Testfall	1	2	3	4
Getestete Äquivalenzklasse	1U, 3U, 4aO, 5U, 5O, 6a-6e	2, 3O	1, 3U	1, 3, 4bU
Gesamtzahl	0	INT_MAX	50	100
VokalAnzahl	0	INT_MAX	0	50
Zchn	'A', 'E', 'I', 'O', 'U', 'Z', '@'	-	-	'I'
Soll-Ergebnisse:				
Gesamtzahl	6	INT_MAX	50	100
VokalAnzahl	5	INT_MAX	0	50



Zufallstest

Füttern des Programms mit *zufälligen Werten*

Einfachstes funktionales Testverfahren

Erreicht Gleichbehandlung aller Eingabedaten ohne Beachtung menschlicher Präferenzen

Bietet sich an in Ergänzung zu anderen Verfahren (z.B. zur Auswahl von zufälligen Repräsentanten aus einer Äquivalenzklasse)

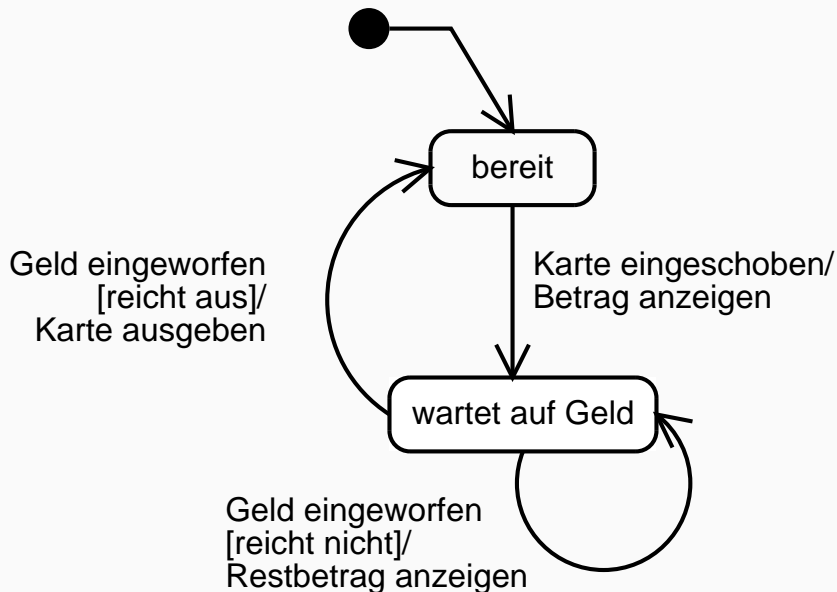




Test von Zustandsautomaten

Ist das Verhalten durch einen *Zustandsautomaten* spezifiziert, können daraus Testfälle abgeleitet werden

Ziel: *Abdeckung aller Zustandsübergänge* (z.B. mindestens 1×)



Beispiel: Parkhaus

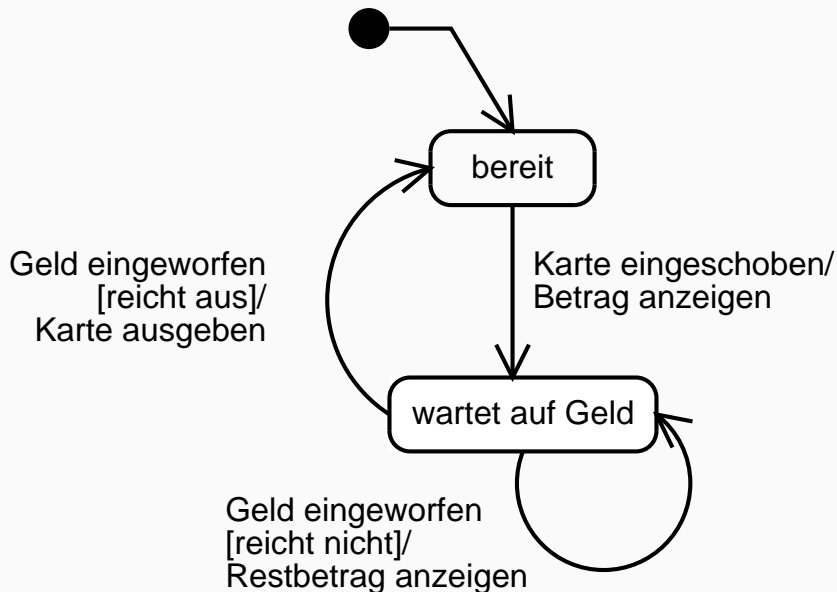




Test von Zustandsautomaten

Ist das Verhalten durch einen *Zustandsautomaten* spezifiziert, können daraus Testfälle abgeleitet werden

Ziel: *Abdeckung aller Zustandsübergänge* (z.B. mindestens 1×)



Beispiel: Parkhaus

Der Testfall

- Karte eingeschoben
- Geld eingeworfen [reicht nicht]
- Geld eingeworfen [reicht aus]

deckt alle Zustände und Übergänge 1× ab





Der Testprozeß

Nur *ein* Testverfahren alleine reicht nicht aus:

- Strukturtests können keine fehlende Funktionalität entdecken
- Funktionstests berücksichtigen die vorliegende Implementierung nur unzureichend (z.B. Zweigabdeckung $< 70\%$)

Daher *kombiniertes Vorgehen*:

1. Funktionstest (mit Grenzwertanalyse + Zufallstests)
2. Strukturtest (der Abschnitte, die noch nicht im Funktionstest abgedeckt wurden)
3. Regressionstest (nach Fehlerkorrektur)





Mutationstesten

Mutationstesten ist ein weiteres Verfahren, die Güte von Testdatensätzen zu bestimmen:

- Annahme: Programme sind „fast richtig“ (*competent programmer hypothesis*)
- Kleine Änderungen am Programm sollten bei hinreichend umfassenden Testdatensätzen zu beobachtbaren Verhaltensänderungen führen
- Die möglichen kleinen Änderungen (*Mutationen*) kann man systematisch erzeugen und die Mutanten mit den Testdatensätzen füttern





Mutationstesten (2)

- Anteil aufgedeckter Mutanten kann als Testgütemaß verwendet werden:
 - „Der Testdatensatz hat nur 60% aller Mutanten gekillt.
Wir brauchen also noch mehr Testfälle“
- Ein perfekter Testdatensatz sollte alle Mutanten erkennen





Mutationsoperatoren

beschreiben Erzeugung semantisch veränderter, aber syntaktisch korrekter Programmversionen

Für jede Fehlerklasse gibt es spezielle Operatoren

Jeder Mutant enthält nur eine Abweichung





Mutationsoperatoren (2)

Berechnungsfehler

Ändern von arithmetischen Operatoren (+ statt –)

Löschen von arithmetischen (Teil-) Ausdrücken

Ändern von Konstanten

Schnittstellenfehler

Vertauschen / Ändern von Parametern

Aufruf anderer Prozeduren eines Moduls

Kontrollflußfehler

Ersetzen von logischen (Teil-) Ausdrücken durch *T* und *F*

Ändern von logischen und relationalen Operatoren

(AND statt OR, \leq statt $<$)

Aufruf anderer Prozeduren

Löschen von Anweisungen





Mutationsoperatoren (3)

Initialisierungsfehler

- Ändern von Konstanten

- Löschen von Zuweisungen / Initialisierungsanweisungen

Datenflußfehler

- Durchtauschen von Variablen in einem Sichtbarkeitsbereich

- Änderungen in der Indexberechnung





Validierung des Testdatensatzes

Bestimmung der Mutantenkillquote zu vorgegebener Mutantenzahl

Schon bei kleiner Mutantenzahl muß ein hoher Prozentsatz Mutanten (>90%) entdeckt werden, ansonsten ist die Testsuite zu klein (= zu wenig Testfälle)





Schätzung der Restfehlerzahl

Gesucht: Restfehlerzahl E

Es werden N Mutationen eingeführt und M Fehler entdeckt

Davon seien X Mutanten. Mithin gilt ungefähr

$$\frac{X}{M} = \frac{N}{E + N}$$

also

$$E = (M - X) \cdot \frac{N}{X}$$

Analogie: Ich setze in einem Teich $N = 100$ markierte Forellen aus. Später fische ich $M = 10$ Forellen; davon sind $X = 2$ markiert. Also schätze ich die Zahl der nicht-markierten Forellen auf $E = (10 - 2) \cdot 100/2 = 400$.

Aber: mit Vorsicht zu genießen, da die Werte stark von N abhängen.





Fazit: Mutationstesten

- ✓ Verfahren zur Beurteilung von Testdatensätzen
- ✓ weitgehend automatisierbar
- ✓ explizite Fehlerorientierung
- ✓ Modellierung anderer Verfahren möglich (z.B. Anweisungsüberdeckung, special value testing)

- ✗ es werden nur „einfache“ Fehler erzeugt
- ✗ es wird angenommen, daß komplexe Fehler Kombinationen einfacher Fehler sind
- ✗ diese Hypothese ist nicht bewiesen





Zusammenfassung

- Die wichtigsten funktionalen Testverfahren sind
 - Funktionale Äquivalenzklassenbildung
 - Grenzwertanalyse
 - Zufallstest
 - Test von Zustandsautomaten
- Da sowohl Funktionstest als auch Strukturtest Nachteile haben, empfiehlt sich ein kombiniertes Vorgehen.
- Mutationstesten hilft, die Güte von Testdatensätzen zu bestimmen



Literatur

- **Lehrbuch der Softwaretechnik, Band 2 (Balzert)**
- **Wissensbasierte Qualitätsassistenz zur Konstruktion von Prüfstrategien für Software-Komponenten (P. Liggesmeyer, 1993)**

