
# Qualitätssicherung von Software

Prof. Dr. Holger Schlingloff

Humboldt-Universität zu Berlin
und
Fraunhofer FIRST

# Kapitel 2. Testverfahren

# Äquivalenzklassenbildung

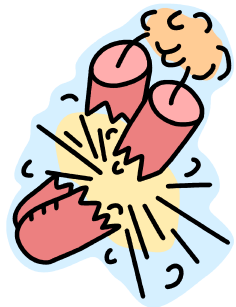
- **1. Schritt:** Partitionierung des Eingabedatenraumes in eine endliche Zahl von Äquivalenzklassen (bezüglich des vermuteten Ausfallverhaltens)
  - im Beispiel: „drei gleiche Eingaben größer Null“
- **2. Schritt:** Auswahl der Testfälle anhand je eines Repräsentanten der Äquivalenzklasse
  - im Beispiel: (2,2,2)

# Vorgehensweise zur Äquivalenzklassenbildung

- Betrachten der Definitionsbereiche für Ein-/Ausgabewerte
- Für jeden Wert ergeben sich gültige und ungültige Klassen
  - Wertebereiche, Aufzählungen: enthalten oder nicht enthalten
  - Eingabewerte, die (möglicherweise) unterschiedlich verarbeitet werden: für jeden Wert eine gültige und insgesamt eine ungültige Klasse
  - Ausgaben, die auf verschiedene Weise berechnet werden: je eine Klasse, die auf diese Ausgabe führt
  - Eingabebedingungen, die vorausgesetzt werden: je eine gültige und eine ungültige Klasse
- Aufspaltung einer Klasse in kleinere Klassen, falls Grund zur Annahme besteht, dass nicht alle Elemente gleich behandelt werden
- Tabellierung der zu jedem Parameter gehörigen Klassen

# Vorgehensweise zur Testfallauswahl

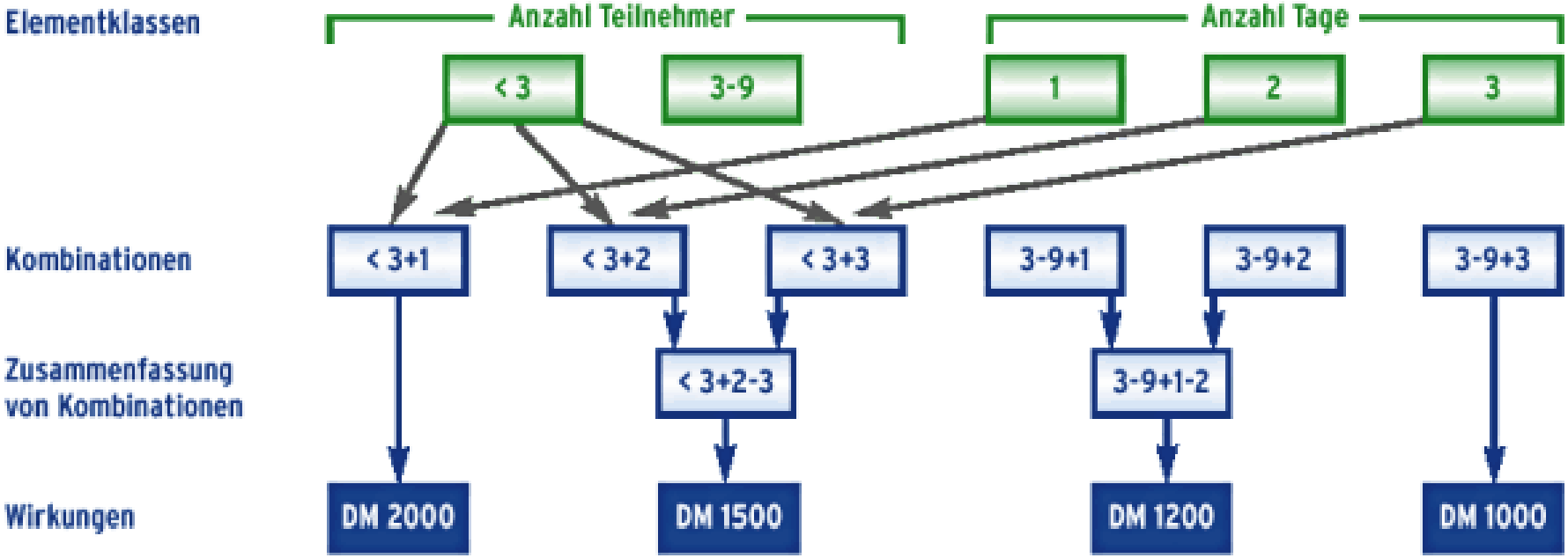
| | Äq1 | Äq2 | Äq3 | ... |
|---|---|---|---|---|
| Par1 | Wert1.1 | Wert1.2 | | |
| Par2 | Wert2.1 | ... | | |
| ... | | | | |
| ParN | | | | |



- Vollständig: Kartesisches Produkt der Klassen
  - meist nicht praktikabel
- Heuristisch: Auswahl gemäß folgender Strategie
  - Bildung von Testfällen, die möglichst viele noch nicht behandelte gültige Klassen abdecken
  - Bildung von Testfällen, die genau eine ungültige Klasse abdecken
  - Paarweise Kombination von Repräsentanten

im Beispiel: (2,2,3) und (-7,1,2), (5,“a“,2)

# Beispiel Äquivalenzklassenmethode




Beispiel aus: SQS-Test Professional
http://www.sqs.de/tools/tool_tcs.htm

# Übungsbeispiel

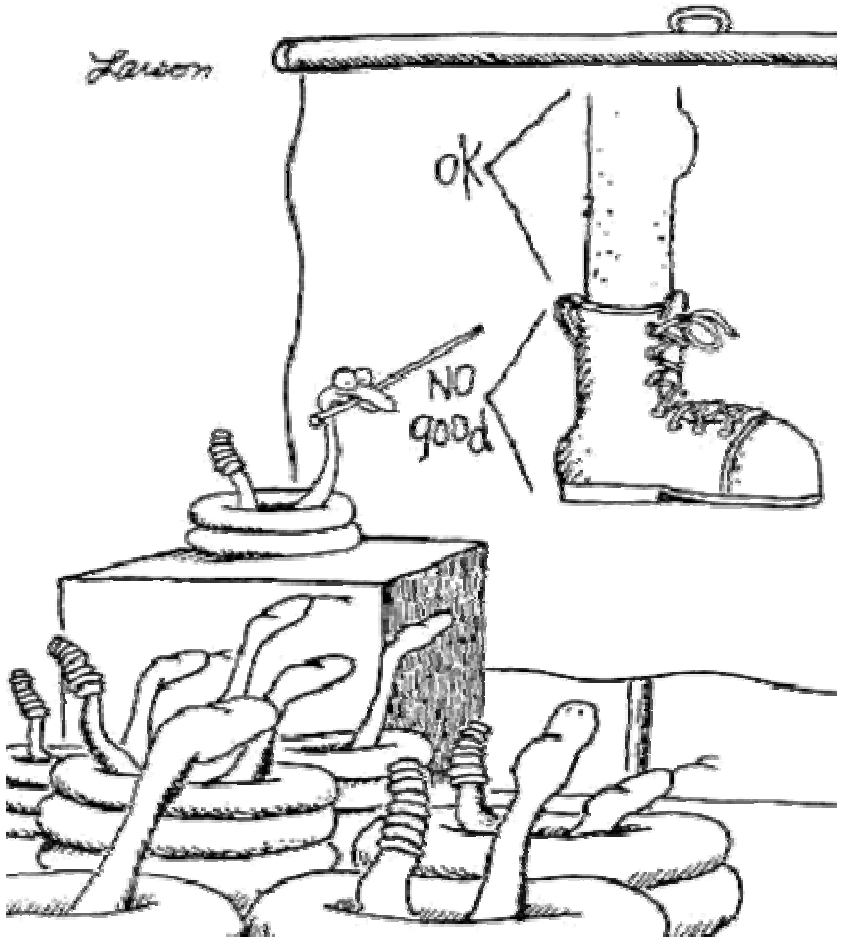
```
public final class IMath {

   public static int idiv (int x, y) {
   /* Returns the integer quotient
      of the two input values */
  ...
   }
}
```

- Bilden Sie Testfälle gemäß der Äquivalenzklassenmethode! (**jetzt!**)
- Welche Fälle wurden nicht erfasst? Warum?

# Äquivalenzklassenmethode – Vor- und Nachteile

- Vorteile
  - systematische Vorgehensweise
  - kalkulierbare Anzahl Testfälle und Überdeckung
  - gut für kleine Funktionen mit Vor- und Nachbedingungen
- Nachteile
  - Testfallauswahl durch Heuristik
  - Wechselwirkungen zwischen Parameterwerten werden oft übersehen
  - bei komplexen Kontrollstrukturen vergleichsweise viele Klassen

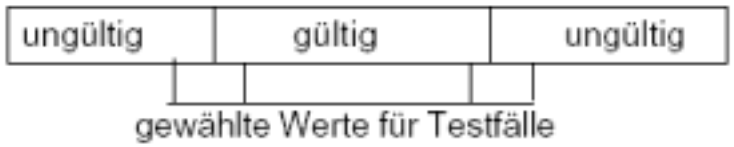
# Pause!




Äquivalenzklassenbildung in der Schlangenschule

# Grenzwertanalyse

- Aufdeckung von Fehlern im Zusammenhang mit der Behandlung der Grenzen von Wertebereichen
- Im Unterschied zur Äquivalenzklassenbildung wird kein beliebiger Repräsentant der Klasse als Testfall ausgewählt, sondern Repräsentanten an den „Rändern“ der Klassen




- In Verzweigungen und Schleifen gibt es oft Grenzwerte, für welche die Bedingung gerade noch zutrifft (oder gerade nicht mehr)
- Rückwärtsanalyse, um Eingabedaten zu erhalten die diese Grenzwerte erreichen

- Grenzwerte werden auf Eingaben und Ausgaben angewendet (! nichttriviales Problem !)
- Testfälle für die Grenzwerte selbst sowie die Werte unmittelbar neben den Grenzwerten
  - Bereiche (atomar oder durch Vorbedingungen festgelegt)
    - Werte auf den Grenzen
    - Werte »rechts bzw. links neben« den Grenzen (ungültige Werte, kleiner bzw. größer als Grenze)
  - Mengen (z.B. bei Eingabedaten, Datenstrukturen, Beziehungen)
    - Kleinste und größte gültige Anzahl
    - Zweitkleinste und zweitgrößte gültige Anzahl
    - Kleinste und größte ungültige Anzahl (oft: leere Menge)
  - Bei strukturierten Daten kartesisches Produkt der Grenzwerte
    - Achtung, kombinatorische Explosion!

nach Fraikin Riedemann Spillner Winter 2004

# Beispiel

- ganzzahlige Eingabe

  { MIN_INT-1,MIN_INT,MIN_INT+1,
  -123,
  -1,0,1,
  456,
  MAX_INT-1,MAX_INT,MAX_INT+1,
  „some-string“ }

- Dateiverarbeitung (80 Zeichen/Zeile)

  Zeile mit 0, 1, 79, 80, 81 Zeichen
  leere Eingabedatei, Datei mit ungültigen Zeichen (eof)

- Rückwärtsanalyse

  public int f (int x){
  ... for (int n = x; n<123; n++){ ... }}

  ➔ 123 ist eine Grenze für x!

- Zusammenfallen der entsprechenden Grenzwerte benachbarter Äquivalenzklassen

# Grenzwertanalyse: Tipps

- Grenzen des Eingabebereichs
  - Bereich: [-1.0;+1.0]; Testdaten: -1.001; -1.0; +1.0; +1.001
  - Bereich: ]-1.0;+1.0[; Testdaten: -1.0; -0.999; +0.999; +1.0
- Grenzen der erlaubten Anzahl von Eingabewerten
  - Eingabedatei mit 1 bis 365 Sätzen; Testfälle 0, 1, 365, 366 Sätze
- Grenzen des Ausgabebereichs
  - Programm errechnet Beitrag, der zwischen 0,00 EUR und 600 EUR liegt; Testfälle: Für 0; 600 EUR und möglichst auch für Beiträge < 0; > 600 EUR
- Grenzen der erlaubten Anzahl von Ausgabewerten
  - Ausgabe von 1 bis 4 Daten; Testfälle: Für 0, 1, 4 und 5 Ausgabewerte
- Erstes und letztes Element bei geordneten Mengen beachten
  - z.B. sequentielle Datei, lineare Liste, Tabelle
- Bei komplexen Datenstrukturen leere Mengen testen
  - z.B. leere Liste, Null-Matrix
- Bei numerischen Berechnungen
  - eng zusammen und weit auseinander liegende Werte wählen

# Grenzwertanalyse: Vor- und Nachteile

- Vorteile
  - An den Grenzen von Äquivalenzklassen sind häufiger Fehler zu finden als innerhalb dieser Klassen
  - "Die Grenzwertanalyse ist bei richtiger Anwendung eine der nützlichsten Methoden für den Testfallentwurf“ (Myers)
  - Effiziente Kombination mit anderen Verfahren, die Freiheitsgrade in der Wahl der Testdaten lassen
- Nachteile
  - Rezepte für die Auswahl von Testdaten schwierig anzugeben
  - Bestimmung aller relevanten Grenzwerte schwierig
  - Kreativität zur Findung erfolgreicher Testdaten gefordert
  - Oft nicht effizient genug angewendet, da sie zu einfach erscheint

# Entscheidungstabellentechnik

- In Entscheidungstabellen werden Variablen mit ihren möglichen Belegungen gelistet sowie die jeweils erwartete Entscheidung
- Typische Anwendungssituation: abhängig von mehreren logischen Eingangswerten sollen verschiedene Aktionen ausgeführt werden
- Jeder Tabellenspalte entspricht ein Testfall
- erzwingen durch ihren Aufbau die logische Vollständigkeit des Tests
- direkt erstellt oder ermittelt aus Ursache-Wirkungs-Graphen

# Ursache-Wirkungs-Graphen

- Zerlegung der Spezifikation in handhabbare Teile
- Ursachen und Wirkung jeder Teilspezifikation ermitteln (Ursache: Einzelne Eingabebedingung oder Äquivalenzklasse von Eingabebedingungen, Wirkung: Ausgabebedingung oder Systemtransformation, Ursachen und Wirkungen werden durch Analyse der Spezifikation ermittelt, jeder Ursache und jeder Wirkung wird eine eindeutige Nummer zugeordnet)
- Transformation der Spezifikation in Ursache-Wirkungs-Graph (und-oder-Kanten)
- Eintragen von Abhängigkeiten zwischen Ursachen und Wirkungen
- Umformung des Graphen in eine Entscheidungstabelle
- Aus jeder Spalte der Entscheidungstabelle einen Testfall erzeugen

# Beispiel

(Fritzsch, http://www.informatik.htw-dresden.de/~fritzsch/QSM/qsm_script.html)

Spezifikation `zaehle`:

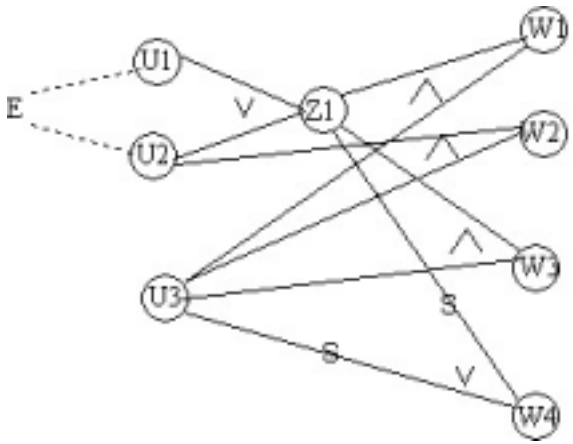
- Die Prozedur `zaehle` liest solange Zeichen, bis ein Zeichen erkannt wird, das kein Großbuchstabe ist, oder `gesamtzahl` den größten `CARDINAL`-Wert erreicht hat.
- ist ein gelesenes Zeichen großer Konsonant oder Vokal, wird `gesamtzahl` um 1 erhöht.
- ist der Großbuchstabe ein Vokal, so wird `vokalanz` um 1 erhöht.
- Bei Beendigung der Prozedur werden `gesamtzahl` und `vokalanz` zurückgegeben

**Ursachen**

U1: `char` ist großer Konsonant
U2: `char` ist großer Vokal
U3: `gesamtzahl` < `max(CARDINAL)`

**Wirkungen**

W1: `gesamtzahl` wird inkrementiert
W2: `vokalanz` wird inkrementiert
W3: `char` wird gelesen
W4: Prozedur wird beendet




| | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Ursachen | U1 | 1 | 0 | 1 | 0 | 0 |
| | U2 | 0 | 1 | 0 | 1 | 0 |
| | U3 | 1 | 1 | 0 | 0 | 1 |
| Wirkungen | W1 | 1 | 1 | 0 | 0 | 0 |
| | W2 | 0 | 1 | 0 | 0 | 0 |
| | W3 | 1 | 1 | 0 | 0 | 0 |
| | W4 | 0 | 0 | 1 | 1 | 1 |

# Implementierungsmöglichkeiten

- Entscheidungstabellen können reduziert werden, da sie redundante Informationen enthalten
- Online-Erstellung der Testfälle aus Entscheidungsdiagramm („BDDs“, viele Forschungsergebnisse im Model Checking)
- Baumartige Darstellung: Entscheidungsbaum

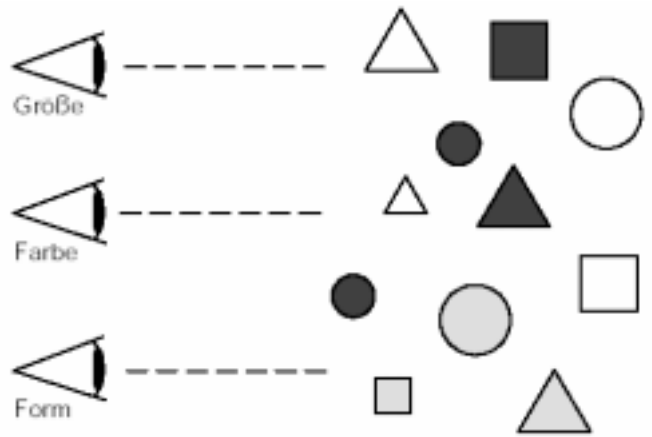
# Klassifikationsbäume

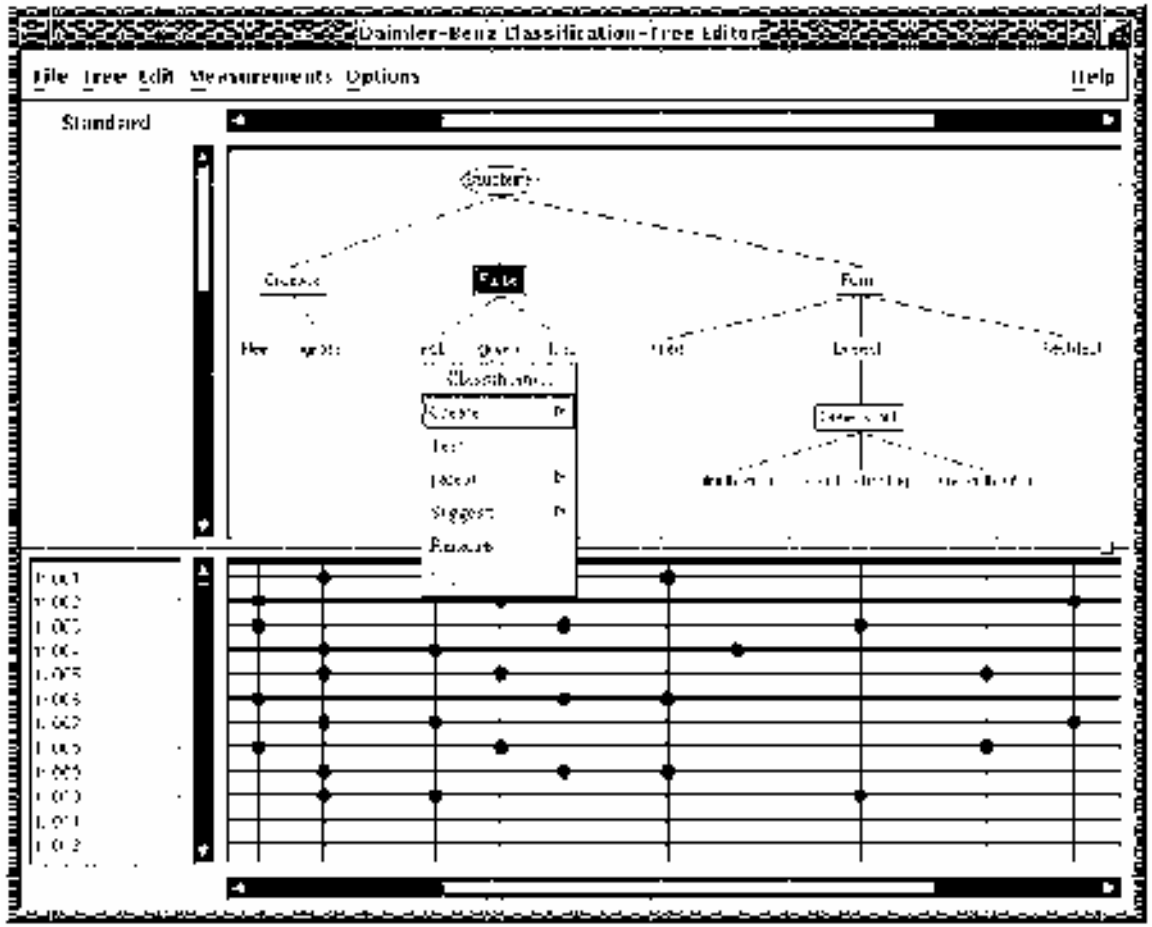
- Entscheidungsbäume sind eine spezielle Darstellungsform von Entscheidungsregeln. Sie veranschaulichen aufeinander folgende hierarchische Entscheidungen
- grundsätzliche Idee der Klassifikationsbaum-Methode ist es, zuerst den Eingabedatenraum des Testobjekts nach verschiedenen testrelevanten Gesichtspunkten jeweils getrennt voneinander in Klassen zu zerlegen, um dann durch die Kombination geeigneter Klassen zu Testfällen zu kommen
- rekursiver Abstieg über mehrere Ebenen ergibt einen Baum von Klassifikationen
- Bildung von Testfällen durch Kombination von Klassen unterschiedlicher Klassifikationen

Literatur: www.systematic-testing.com

# Beispiel Klassifikationsbaum-Methode

- Computer-Vision: Erkennung von Objekten




- Erweiterungen für kontinuierliche Werteverläufe
- automatische Testwerkzeuge

Beispiel nach Wegener/Grochtmann, DaimlerChrysler AG

# Modultest: Black or white?

- Black-box Tests

  betrachten die zu untersuchende Komponente als uneinsehbare Einheit und beobachten sie nur an den freigegebenen nach außen sichtbaren Schnittstellen

- White-box oder glass-box Tests

  erlauben dem Tester Zugriff auf Programmstruktur und evtl. auf interne Variablen der zu testenden Komponente

werden im Kapitel Überdeckungsmaße behandelt