LE 15
1

Software-Qualitätssicherung

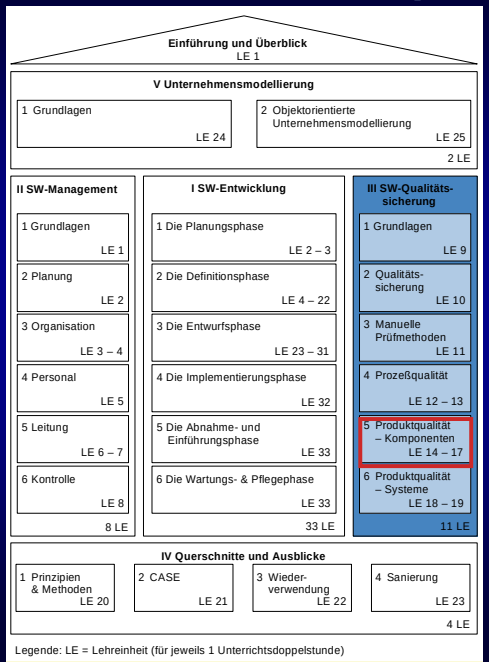
5 Produktqualität – Komponenten: Testende Verfahren 2

[stark gekürzt]

Prof. Dr. Helmut Balzert
Lehrstuhl für Software-Technik
Ruhr-Universität Bochum

© Helmut Balzert 1998

LE 15
2



LE 15
3

Inhalt

5.5 Datenflußorientierte Strukturtestverfahren
- 5.5.1 Die *Defs/Uses*-Verfahren
- 5.5.2 Weitere Verfahren

5.6 Funktionale Testverfahren
- 5.6.1 Funktionale Äquivalenzklassenbildung
- 5.6.2 Grenzwertanalyse
- 5.6.3 Test spezieller Werte
- 5.6.4 Zufallstest
- 5.6.5 Test von Zustandsautomaten

5.7 Kombinierter Funktions- und Strukturtest.

LE 15
4

5.5 Datenflußorientierte Strukturtestverfahren

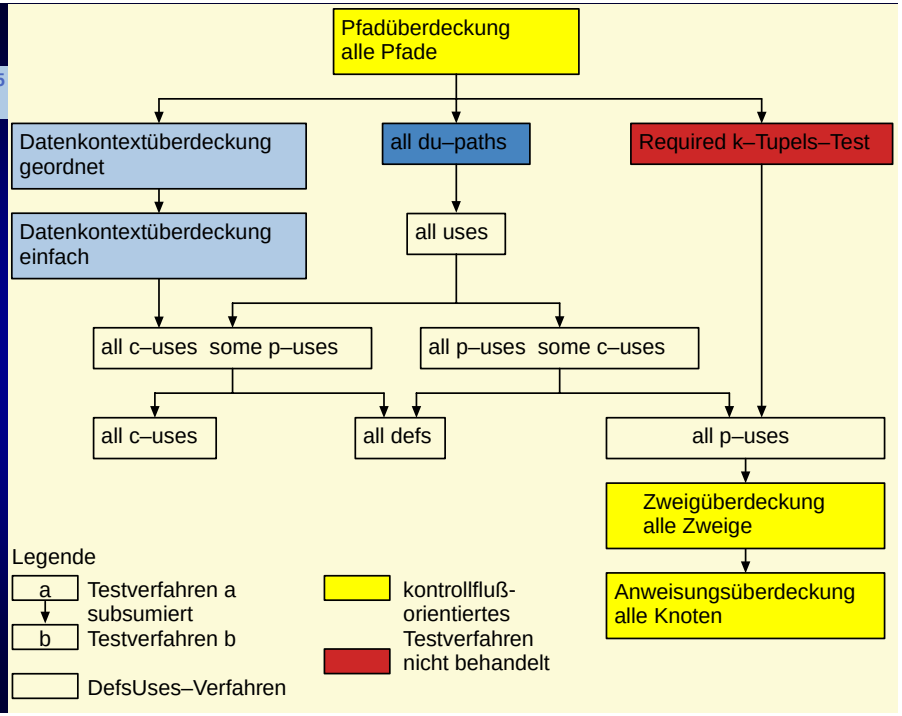
- **Einordnung**
  - Verwenden zur Erzeugung von Testfällen die **Zugriffe auf Variablen**
  - Gehören zu den **Strukturtestverfahren**, da die Programmstruktur sichtbar sein muß
  - Gehören zu den **dynamischen Testverfahren**, da das Programm beim Testen ausgeführt wird
- **Testziele durch...**
  - Definition von Variablen
  - lesende und schreibende Zugriffe auf Variablen in Anweisungen und Bedingungen.

LE 15
5

5.5 Datenflußorientierte Strukturtestverfahren

- **Einsatzbereich**
  - **Test von:**
    - **Datenobjektmodulen**
    - **Datentypmodulen**
    - **Klassen**
- **Beschränkungen**
  - **Nur wenige Testwerkzeuge vorhanden**
    - **Es werden nur die *Defs/Uses*-Verfahren behandelt.**

LE 15
6



LE 15
7

5.5.1 Die *Defs/Uses*-Verfahren

- ***Defs/Uses*-Verfahren**
  - **Testfälle werden auf der Basis von Variablen-zugriffen definiert**
  - **Variablenzugriffe dienen entweder:**
    - **Zur Wertzuweisung – auch Definition *(def)* genannt**
    - **Zur Berechnung von Werten innerhalb eines Ausdrucks *(computational-use, c-use*, berechnende Benutzung)**
    - **Zur Bildung von Wahrheitswerten in Bedingungen bzw. Prädikaten *(predicate-use, p-use*, prädikative Benutzung).**

LE 15
8

5.5.1 Die *Defs/Uses*-Verfahren

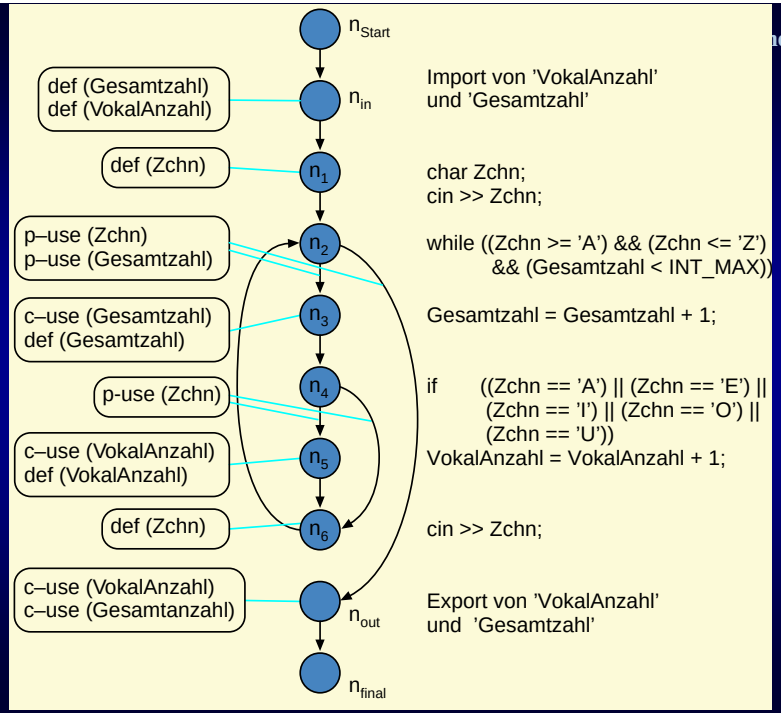
- **Beispiel**
  - **Die Anweisung `y=x+1` enthält eine berechnende Referenz der Variablen `x` gefolgt von einer Wertzuweisung an die Variable `y`**
  - **Die Bedingung `if(x>5)...` enthält eine prädikative Referenz der Variablen `x`**
- **Modifizierte Form des Kontrollflußgraphen**
  - **Unterschiede zur üblichen Grundform:**
    - **Zuordnung von Datenflußattributen zu den Knoten und Kanten**
    - **Existenz der zusätzlichen Knoten n_{in} und n_{out} in der Datenflußdarstellung**
    - **Beschreiben Informationsimport/-export über Parameter oder globale Variablen.**

LE 15
9

5.5.1 Die *Defs/Uses*-Verfahren

- Wertzuweisungen an und Benutzungen von Variablen sind ausschließlich in den Knoten des Kontrollflußgraphen möglich
- Bedingungen werden mit den Kanten des Kontrollflußgraphen assoziiert
- Ist die letzte Anweisung eines Knotens eine Kontrollstruktur mit der Bedingung $p(x_1,..,x_n)$, enthalten alle von diesem Knoten ausgehenden Kanten prädikative Benutzungen von $x_1, ..., x_n$.

LE 15
10



LE 15
11

5.6 Funktionale Testverfahren

- **Funktionale Testverfahren** *(Black Box)*
  - Testfälle aus der **Programmspezifikation** ableiten
  - Programmstruktur wird **nicht** betrachtet
    - Sollte für den Tester unsichtbar sein
    - Testling sollte für den Tester mit Ausnahme der Spezifikation ein »schwarzer Kasten« sein
  - Begründung, ein Programm gegen seine Spezifikation zu testen:
    - Es ist unzureichend, ein Programm lediglich gegen sich selbst zu testen
  - Aufgabe eines Programms:
    - Transformation von Eingaben in Ausgaben
    - Diese Funktionalität ist in der Spezifikation beschrieben.

LE 15
12

5.6 Funktionale Testverfahren

- Ziel
  - Möglichst umfassende, aber redundanzarme Prüfung der spezifizierten Funktionalität
  - Überprüfung aller Programmfunktionen notwendig
  - Testfälle und Testdaten werden allein aus der Programmspezifikation abgeleitet
  - Funktionsüberdeckung.

LE 15
13

5.6 Funktionale Testverfahren

- Programmspezifikation
  - Notwendig zur Beurteilung
    - der Vollständigkeit eines Funktionstests
    - der Korrektheit der erzeugten Ausgaben
  - Notwendig zur Definition von Testfällen
  - Im Idealfall liegt sie in formaler Form vor
  - Üblicherweise existieren jedoch nur...
    - semiformale oder
    - informale Spezifikationen.

LE 15
14

5.6 Funktionale Testverfahren

- Aufgabe der Testplanung
  - Aus der Programmspezifikation Testfälle herzuleiten, mit denen das Programm getestet werden soll
  - Zu einem Testfall gehören
    - Eingabedaten in das Testobjekt
    - Erwartete Ausgabedaten oder Ausgabereaktionen (Soll-Ergebnisse)
- Hauptschwierigkeiten
  - Ableitung der geeigneten Testfälle
  - Vollständiger Funktionstest ist i. allg. nicht durchführbar.

LE 15
15

5.6 Funktionale Testverfahren

- Ziel einer Testplanung
  - Testfälle so auswählen, daß die Wahrscheinlichkeit groß ist, Fehler zu finden
- Testfallbestimmung:
  - Funktionale Äquivalenzklassenbildung
  - Grenzwertanalyse
  - Test spezieller Werte
  - Zufallstest
  - Test von Zustandsautomaten.

LE 15
16

Funktionale Äquivalenzklassenbildung (1/5)

- Einordnung
  - Dynamisches, funktionales Testverfahren, das Tests aus der funktionalen Spezifikation eines Programms ableitet
  - Dies geschieht durch Bildung von Äquivalenzklassen
- Ziel
  - Die Definitionsbereiche der Eingabeparameter und die Wertebereiche der Ausgabeparameter werden in Äquivalenzklassen zerlegt
  - Es wird davon ausgegangen, daß ein Programm bei der Verarbeitung eines Repräsentanten aus einer Äquivalenzklasse so reagiert, wie bei allen anderen Werten aus dieser Äquivalenzklasse.

LE 15
17

Funktionale Äquivalenzklassenbildung (2/5)

- Läuft das Programm mit dem repräsentativen Wert der Äquivalenzklasse fehlerfrei, ist zu erwarten, daß es auch für andere Werte aus dieser Äquivalenzklasse korrekt funktioniert
- Voraussetzung:
  - Sorgfältige Wahl der Äquivalenzklassen
- Äquivalenzklassen können sich überschneiden
  - Äquivalenzklassenbegriff nicht im strengen Sinne der Mathematik zu verstehen.

LE 15
18

Funktionale Äquivalenzklassenbildung (3/5)

- Beispiel

```
void setzeMonat(short aktuellerMonat);
                                //Eingabeparameter
//Es muß gelten: 1 ≤ aktuellerMonat ≤ 12
```

- 1 gültige Äquivalenzklasse:
  - `1 ≤ aktuellerMonat ≤ 12`
- 2 ungültige Äquivalenzklassen:
  - `aktuellerMonat < 1, aktuellerMonat > 12`
- Aus den Äquivalenzklassen abgeleitete Testfälle:

  1 `aktuellerMonat = 5`
  (Repräsentant der Äquivalenzklasse)

  2 `aktuellerMonat = -3`

  3 `aktuellerMonat = 25.`

LE 15
19

Funktionale Äquivalenzklassenbildung (4/5)

- Eigenschaften
  - Äquivalenzklassen der Eingabewerte bilden
    - Alle spezifizierten Funktionen mit Werten aus der ihnen zugeordneten Äquivalenzklasse testen
    - Dadurch Beschränkung der Testfallanzahl
  - Äquivalenzklassen aus den Ausgabewertebereichen erstellen
    - Für diese Eingaben finden, die die gewünschten Ausgabewerte erzeugen
    - Die Ausgabeäquivalenzklassen müssen in entsprechende Äquivalenzklassen von Eingabewerten umgeformt werden
  - Repräsentant für einen Testfall
    - Irgendein Element aus der Klasse auswählen.

LE 15
20

Funktionale Äquivalenzklassenbildung (5/5)

- Bewertung
  - Geeignetes Verfahren, um aus Spezifikationen – insbesondere aus Parameterein- und -ausgaben – repräsentative Testfälle abzuleiten
  - Basis für die Grenzwertanalyse
  - Die Aufteilung in Äquivalenzklassen muß nicht mit der internen Programmstruktur übereinstimmen
  - Es werden einzelne Eingaben oder Ausgaben betrachtet
  - Beziehungen, Wechselwirkungen und Abhängigkeiten zwischen Werten werden nicht behandelt
    - Dazu werden Verfahren wie die Ursache-Wirkungs-Analyse benötigt.

LE 15
21

5.6.1 Funktionale Äquivalenzklassenbildung

- Beispiel 1a
  - Spezifikation von ZaehleZchn:
    - Aufgabe:
      Die Prozedur `ZaehleZchn` liest solange Zeichen von der Tastatur (Eingabevariable `Zchn`, Typ `char`), bis ein Zeichen erkannt wird, das kein Großbuchstabe ist, oder bis `Gesamtzahl` den größten durch den Datentyp `int` darstellbaren Wert `INT_MAX` erreicht
      - Ist ein gelesenes Zeichen ein Großbuchstabe zwischen A und Z, wird `Gesamtzahl` um eins erhöht
      - Ist der Großbuchstabe ein Vokal, dann wird auch `VokalAnzahl` um eins erhöht
      - Ein-/Ausgabeparameter sind `Gesamtzahl` und `VokalAnzahl`.

LE 15
22

5.6.1 Funktionale Äquivalenzklassenbildung

- Randbedingung:
  - Das aufrufende Programm stellt sicher, daß `Gesamtzahl` stets größer oder gleich `VokalAnzahl` ist
- `void ZaehleZchn(int &VokalAnzahl, int &Gesamtzahl);`
  - Anhand der Spezifikation sieht man, daß der Testling 2 E/A-Parameter und 1 interaktive Eingabe besitzt
  - Die 2 E/A-Parameter `VokalAnzahl` und `Gesamtzahl` sind vom Datentyp `int`
  - Außerdem muß `Gesamtzahl <= INT_MAX` sein
  - Randbedingung: `VokalAnzahl ≤ Gesamtzahl.`

LE 15
23

5.6.1 Funktionale Äquivalenzklassenbildung

- Äquivalenzklassen

  1 `0 <= Gesamtzahl < INT_MAX`

  2 `Gesamtzahl = INT_MAX`

  Für `VokalAnzahl` ergibt sich folgende Äquivalenzklasse:

  3 `0 ≤ VokalAnzahl < Gesamtzahl.`

LE 15
24

5.6.1 Funktionale Äquivalenzklassenbildung

- Die interaktive Eingabevariable `Zchn` ist vom Typ `char`, d.h. alle Zeichen sind zugelassen
- Die Spezifikation sagt jedoch, daß sich das Programm für folgende Fälle offenbar unterschiedlich verhält:

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

5 `Zchn` ist ein Großbuchstabe

6 `Zchn` ist ein großer Vokal
`('A','E','I','O','U')`.

LE 15
25

5.6.1 Funktionale Äquivalenzklassenbildung

- Äquivalenzklasse 4 besteht im Zeichensatz aus 2 zusammenhängenden Teilen, die wahrscheinlich unterschiedlich behandelt werden:

  4a `Zchn < 'A'`

  4b `Zchn > 'Z'`
- Äquivalenzklasse 5 umfaßt den geordneten Bereich `'A' < Zchn < 'Z'`
- Äquivalenzklasse 6 enthält nur die 5 Vokale
  - Es ist zu erwarten ist, daß sie von der Prozedur unterschiedlich behandelt werden
  - Aufgrund der geringen Anzahl der Elemente:
    - Aufteilung in 5 Klassen mit jeweils einem Element
- Nach Identifikation der Äquivalenzklassen:
  - Repräsentanten auswählen
  - Anschl. Testfälle zusammenstellen.

LE 15
26

5.6.1 Funktionale Äquivalenzklassenbildung

- Testfälle für `ZaehleZchn`

Testfall	1	2	3
getestete Äquivalenzklasse	1, 3, 5, 6a, 6b, 6c, 6d, 6e, 4a	1, 3, 4b	2, 3
Gesamtzahl	100	1	INT_MAX
VokalAnzahl	50	1	50
Zchn	'X', 'A', 'E', 'I', 'O', 'U', 'I'	'a'	–
Soll–Ergebnisse: Gesamtzahl	106	1	INT_MAX
VokalAnzahl	55	1	50

LE 15
27

5.6.1 Funktionale Äquivalenzklassenbildung

- Testfälle für ungültige Äquivalenzklassen
  - Werden durch Auswahl eines Testdatums aus einer ungültigen Äquivalenzklasse gebildet
  - Es wird mit Werten kombiniert, die ausschließlich aus gültigen Äquivalenzklassen entnommen sind
  - Da für alle ungültigen Eingabewerte eine Fehlerbehandlung existieren muß, kann bei Eingabe eines fehlerhaften Wertes pro Testfall die Fehlerbehandlung nur durch dieses fehlerhafte Testdatum verursacht worden sein
  - Würden mehrere fehlerhafte Eingaben pro Testfall verwendet, so ist nicht transparent, welches fehlerhafte Testdatum die Fehlerbehandlung ausgelöst hat.

LE 15
28

Grenzwertanalyse (1/4)

- Einordnung
  - Dynamisches, funktionales Testverfahren
  - Leitet Tests aus der funktionalen Spezifikation eines Programms ab
  - Basiert in der Regel auf der funktionalen Äquivalenzklassenbildung
  - Gehört ebenfalls in die Kategorie »Test spezieller Werte«
- Ziel
  - Testfälle, die die Grenzwerte der Äquivalenzklassen abdecken oder in der unmittelbaren Umgebung dieser Grenzen liegen, decken besonders häufig Fehler auf.

LE 15
29

Grenzwertanalyse (2/4)

- Es wird *nicht* irgendein Element aus der Äquivalenzklasse als Repräsentant ausgewählt
- Es werden ein oder mehrere Elemente ausgesucht, so daß jeder Rand der Äquivalenzklasse getestet wird
- Annäherung an die Grenzen der Äquivalenzklasse
  - Vom gültigen Bereich aus
  - Vom ungültigen Bereich aus.

LE 15
30

Grenzwertanalyse (3/4)

- Beispiel

```
void setzeMonat (short aktuellerMonat);
                                //Eingabeparameter
```

  - Es muß gelten: $1 \leq$ aktuellerMonat ≤ 12
  - 1 gültige Äquivalenzklasse: $1 \leq$ aktuellerMonat ≤ 12
  - 2 ungültige Äquivalenzklassen: aktuellerMonat < 1, aktuellerMonat > 12
  - Abgeleitete Testfälle

    1 aktuellerMonat = 1 (untere Grenze)

    2 aktuellerMonat = 12 (obere Grenze)

    3 aktuellerMonat = 0 (obere Grenze der ungültigen Äquivalenzklasse)

    4 aktuellerMonat = 13 (untere Grenze der ungültigen Äquivalenzklasse).

LE 15
31

Grenzwertanalyse (4/4)

- Eigenschaften
  - Grenzwertanalyse nur dann sinnvoll
    - Wenn die Menge der Elemente, die in eine Äquivalenzklasse fallen, auf natürliche Weise geordnet werden kann
  - Bei Äquivalenzklassenbildung können wichtige Typen von Testfällen Übersehen werden
- Bewertung
  - Fehlerbasiertes Kriterium, da ihr die Erfahrung, daß insbesondere Grenzbereiche häufig fehlerhaft verarbeitet werden, zugrundeliegt
  - Sinnvolle Erweiterung und Verbesserung der funktionalen Äquivalenzklassenbildung.

LE 15
32

5.6.3 Test spezieller Werte

- Test spezieller Werte
  - Zusammenfassung einer Reihe von Testverfahren
  - Fordern für die Eingabedaten selbst, oder für von den Eingabedaten abhängige Aspekte bestimmte Eigenschaften
- Ziel
  - Aus der Erfahrung heraus fehlersensitive Testfälle aufzustellen
  - Grundidee, eine Liste möglicher Fehler oder Fehlersituationen aufzustellen und daraus Testfälle abzuleiten
  - Zusammengestellung von Spezialfällen, die unter Umständen auch bei der Spezifikation übersehen wurden.

LE 15
33

5.6.3 Test spezieller Werte

- Einordnung
  - Die Testfälle werden nicht unbedingt aus der Spezifikation abgeleitet
  - Oft in Kombination mit anderen Teststrategien, z.B. der Äquivalenzklassenbildung, eingesetzt
- Kriterien
  - Für die Auswahl von Testdaten sind viele Kriterien denkbar, abhängig von bestimmten Fehlererwartungshaltungen:
    - Die Grenzwertanalyse
    - Das *zero values*-Kriterium
    - Das *distinct values*-Kriterium.

LE 15
34

5.6.3 Test spezieller Werte

- *zero values*-Kriterium
  - Tests, die Null an Variablen zuweisen
- *distinct values*-Kriterium
  - Zuweisung unterschiedlicher Werte an Feldelemente und Eingabedaten, die miteinander in Beziehung stehen
- Beispiele für spezielle Testwerte sind:
  - Wert 0 als Eingabe- oder Ausgabewert
    - Zeigt oft eine fehlerhafte Situation an
  - Eingabe von Zeichenketten
    - Sonderzeichen oder Steuerzeichen fehleranfällig
  - Tabellenverarbeitung
    - »kein Eintrag« und »ein Eintrag« Sonderfälle.

LE 15
35

5.6.4 Zufallstest

- Zufallstests
  - Aus dem Wertebereich der Eingabedaten werden zufällig Testfälle erzeugt
- Voraussetzungen
  - Datentypen und Datenstrukturen der Eingabedaten müssen bekannt sein
    - Entweder aus der Spezifikation oder der Implementierung des Prüflings entnehmen
- Werkzeug
  - Generiert entsprechende Testdaten.

LE 15
36

Test von Zustandsautomaten (1/4)

- Einordnung
  - Dynamisches, funktionales Testverfahren, das Tests aus der in Form eines Zustandsautomaten vorliegenden Spezifikation eines Programms ableitet
- Ziel
  - Durch Testfälle Überdeckung aller Zustands-übergänge, d.h. das mindestens einmalige Durchlaufen aller Zustandsübergänge
  - Minimale Teststrategie:
    - Mindestens einmalige Abdeckung aller Zustände durch Testfälle.

- Danke!
- Aufgaben

- Diese Präsentation bzw. Teile dieser Präsentation enthalten Inhalte und Grafiken des Lehrbuchs der Software-Technik (Band 2) von Helmut Balzert, Spektrum Akademischer Verlag, Heidelberg 1998