Fachgruppe Praktische Informatik

Verifikation, Validation und Testen sicherheitskritischer Systeme

Ausarbeitung im Rahmen des Seminars

Sicherheitskritische Systeme

von

Chandra Kurnia Jaya

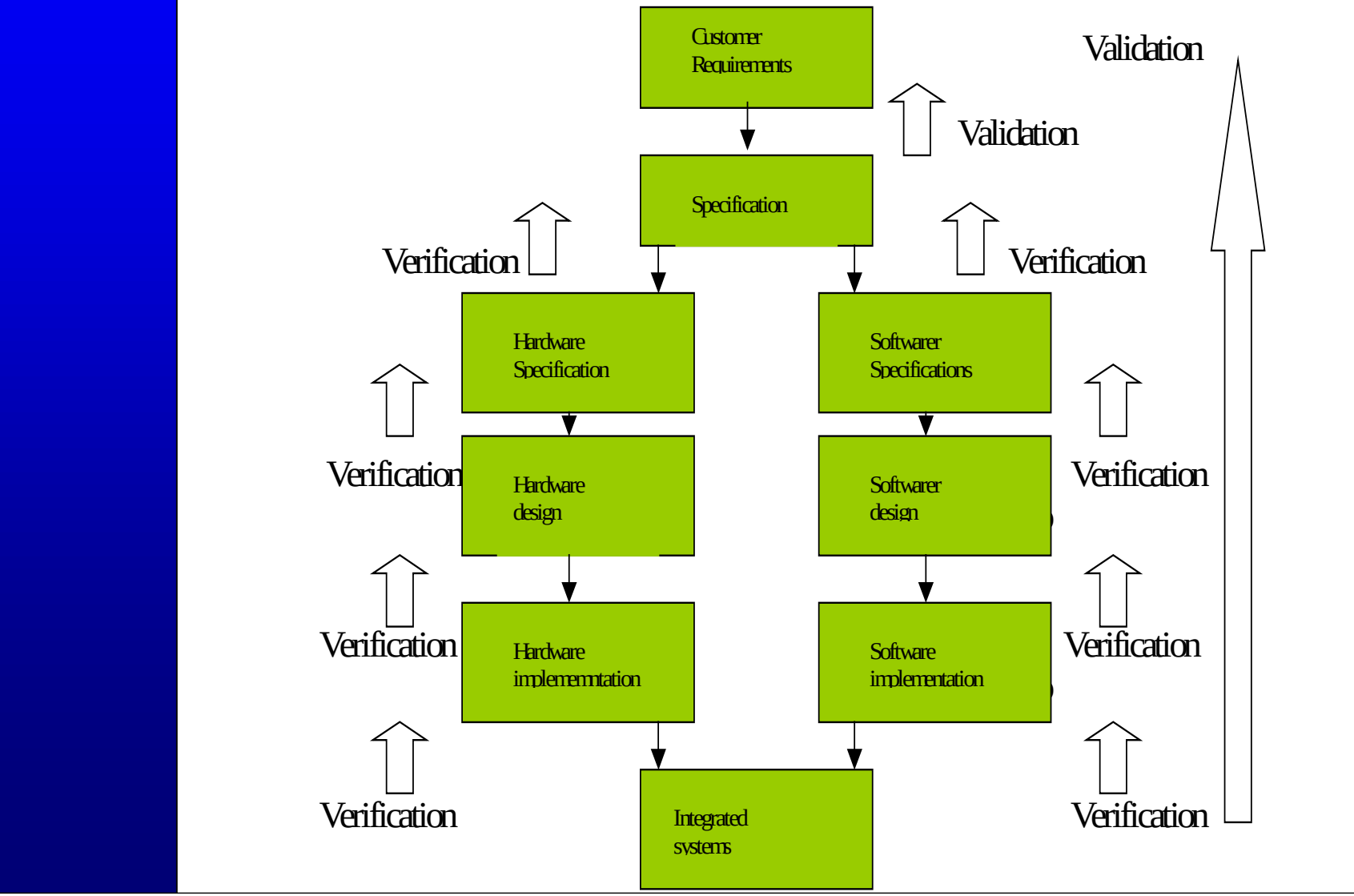
betreut durch

Dr. Jörg Niere

Definition (1)

- Sicherheitskritische Software
- Verifikation
- Validation
- Spezifikation
- Testen

Definition (2)



Unterschied zwischen Verification & Validation

- Verifikation
  - kann nicht getestet werden
  - Beweis mit der formalen Verifikation : Model Checking und Theorem Proving

- Validation
  - Die zentrale Tätigkeit ist das Testen
  - Zwei Strategie für Testen : Black-Box-Test und White-Box-Test

Testen : Black-Box-Test (1)

- Was ist Black-Box-Test ?
- Drei wichtige Verfahren :
  - Äquivalenzklassenbildung (Equivalence Partitioning)
  - Grenzwertanalyse (Boundary Value Analysis)
  - Test spezieller Werte (Error-Guessing)
- Vorteile und Nacteile von Black-Box-Test

Testen : Black-Box-Test (2)

Was ist Black-Box-Test ?

- Tester kennt nur “was eine Funktion macht“
- Ableitung der Testfälle aus der Spezifikation, Benutzerhandbuch.
- Testfälle für möglichst viele gültige Eingabe
- Testfälle für möglichst viele ungültige Eingabe

Testen : Black-Box-Test (3)

Definition von Äquivalenzklassenbildung

- Menge von Eingabedaten, die auf ein Programm eine gleichartige Wirkung ausüben.
- Es werden Äquivalenzklasse für gültige und ungültige Daten gebildet
- Basis für die Grenzwertanalyse

Testen : Black-Box-Test (5)

Beispiel von Äquivalenzklassenbildung

Eingabe : vertragsbeginn, geburtsdatum	
Hilsvariable : 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

Spezifikation zur Ableitung des technisches Eintrittsalter einer Person in einen Versicherungsvertrag.

Testen : Black-Box-Test (6)

Beispiel von Äquivalenzklassenbildung

Testfall			Ausgewähltes Testdatum		
	Äquivalenzklasse	Ausgabe	Geburts-datum	Vertrags-beginn	Ausgabe :soll
T1	1 Vertragsbeginn vor Geburtsdatum	Fehler	01.02.2001	01.01.2001	Fehler
T2	2 diff_Monat im Interval [-5, 6]	diff_Jahr	01.06.1975	01.08.2001	26
T3	3 diff_Monat > 6	diff_Jahr+1	01.05.1975	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 sind Tag, Monat, Jahr ,die nicht im gültigen Wertebereich liegt (Zum Beispiel : negative Zahlen).

Testen : Black-Box-Test (7)

Bildung von Äquivalenzklassenbildung

- Einen zusammenhängenden Wertebereich -->
  eine gültige und zwei ungültige ÄK
- Eine Anzahl von Werten --> eine gültige und zwei ungültige ÄK
- Eine Situation, die zwingend erfüllt sein muss-->
  eine gültige und eine ungültige ÄK
- Falls die Eingangbedingung als Menge vom gültigen Eingabewert
- Werden die Werte einer ÄK unterschiedlich behandelt, so teilt man diese ÄK auf

Testen : Black-Box-Test (8)

Vorteile und Nachteile von Äquivalenzklassenbildung

- Vorteile von der Äquivalenzklassenbildung :

  1.Äquivalenzklassenbildung ist das Basis für die Grenzwertanalyse.

  2.Äquivalenzklassenbildung ist ein geeignetes Verfahren, um aus Spezifikationen repräsentative Testfälle abzuleiten

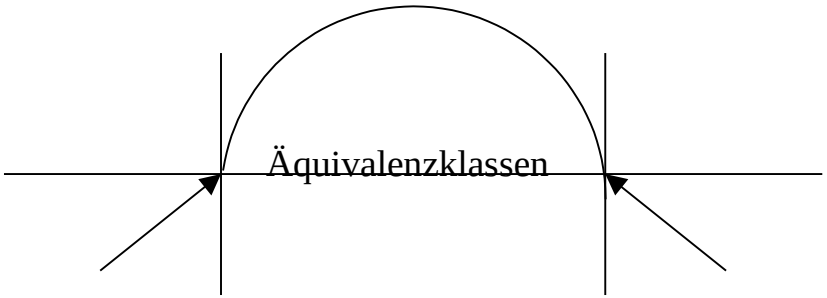
- Nachteile von der Äquivalenzklassenbildung

  1. Es wird nur einzelne Eingaben betrachtet. Beziehungen, Wechselwirkungen zwischen Werten werden nicht behandelt

Testen : Black-Box-Test (9)

Grenzwertanalyse

- Basiert auf der funktionalen Äquivalenzklassenbildung
- Grenzwertanalyse benutzt Testdaten von Äquivalenzklassen, welche nur die Werte an den Rändern berücksichtigt werden
- dieses Verfahren sehr oft benutzt, um den Fehler an den Rändern zu entdecken.



Testen : Black-Box-Test (10)

Beispiel von Grenzwertanalyse

Eingabe : vertragsbeginn, geburtsdatum	
Hilsvariable : 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

Spezifikation zur Ableitung des technisches Eintrittsalter einer Person in einen Versicherungsvertrag.

Testen : Black-Box-Test (11)

Beispiel von Grenzwertanalyse

Testfall			Ausgewähltes Testdatum		
	Eingabe	Ausgabe	Geburts-datum	Vertrags-beginn	Ausgabe :soll
T1-2	Vertragsbeginn 1 Tag vor Geburtsdatum	Fehler	02.02.2001	01.02.2001	Fehler
T2-1	Vertragsbeginn = Geburtsdatum	0	01.06.1975	01.06.1975	0
T2-3	diff_Monat = 6	diff_Jahr	01.06.1975	01.12.2001	26
T2-4	diff_Monat = -5	diff_Jahr	01.06.1975	01.01.2001	26
T3-2	diff_Monat = 7	diff_Jahr+1	01.05.1975	01.12.2001	27
T4-2	diff_Monat = -6	diff_Jahr-1	01.07.1975	01.01.2001	25

Wir können leicht verstehen, dass wir hier diff_Monat = -5, -6, 6, 7 für den Grenzwert benutzen . Ausserdem wird hier untersucht , wie sich das Programm bei falschen Werten (hier Vertragsbeginn 1 Tag vor Geburtsdatum) verhält

Testen : Black-Box-Test (12)

Vorteile und Nachteile von Grenzwertanalyse

- Vorteile von der Grenzwertanalyse

  1. Erweiterung und Verbesserung der Äquivalenzklassenbildung
  2. Fehler an den Rändern werden häufig entdeckt

- Nachteile von der Grenzwertanalyse

  1. Es ist schwierig, Rezepte für die Grenzwertanalyse zu geben, denn dieses Verfahren erfordert die Kreativität vom Tester für das gegebene Problem
  2. Es wird nur einzelne Eingaben betrachtet. Beziehungen, Wechselwirkungen zwischen Werten werden nicht behandelt

Testen : Black-Box-Test (13)

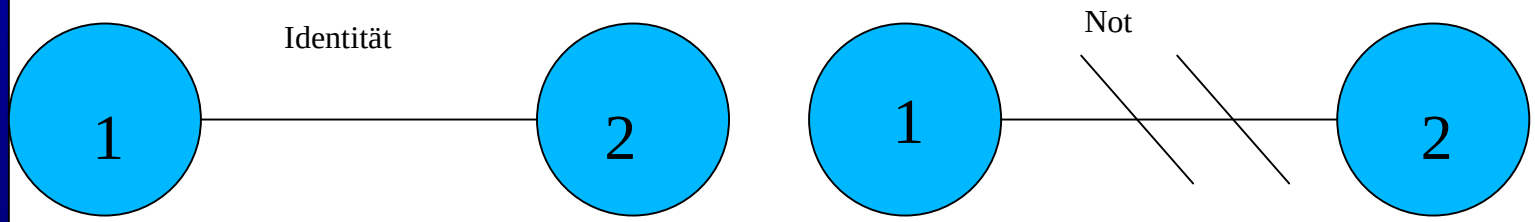
Test spezieller Werte (Error-Guessing)

- Das Error-Guessing ist in der eigentlichen Sinne keine Testmethode
- Diese Methode beruht auf der Erfahrung und Wissen des Testers
- Prinzipiell legt man eine Liste möglicher Fehler oder fehleranfälliger Situationen an und definiert damit die neuen Testfälle.
- Beispiel : Der Wert 0 als Eingabewert zeigt oft eine fehleranfällige Situation

Testen : Black-Box-Test (14)

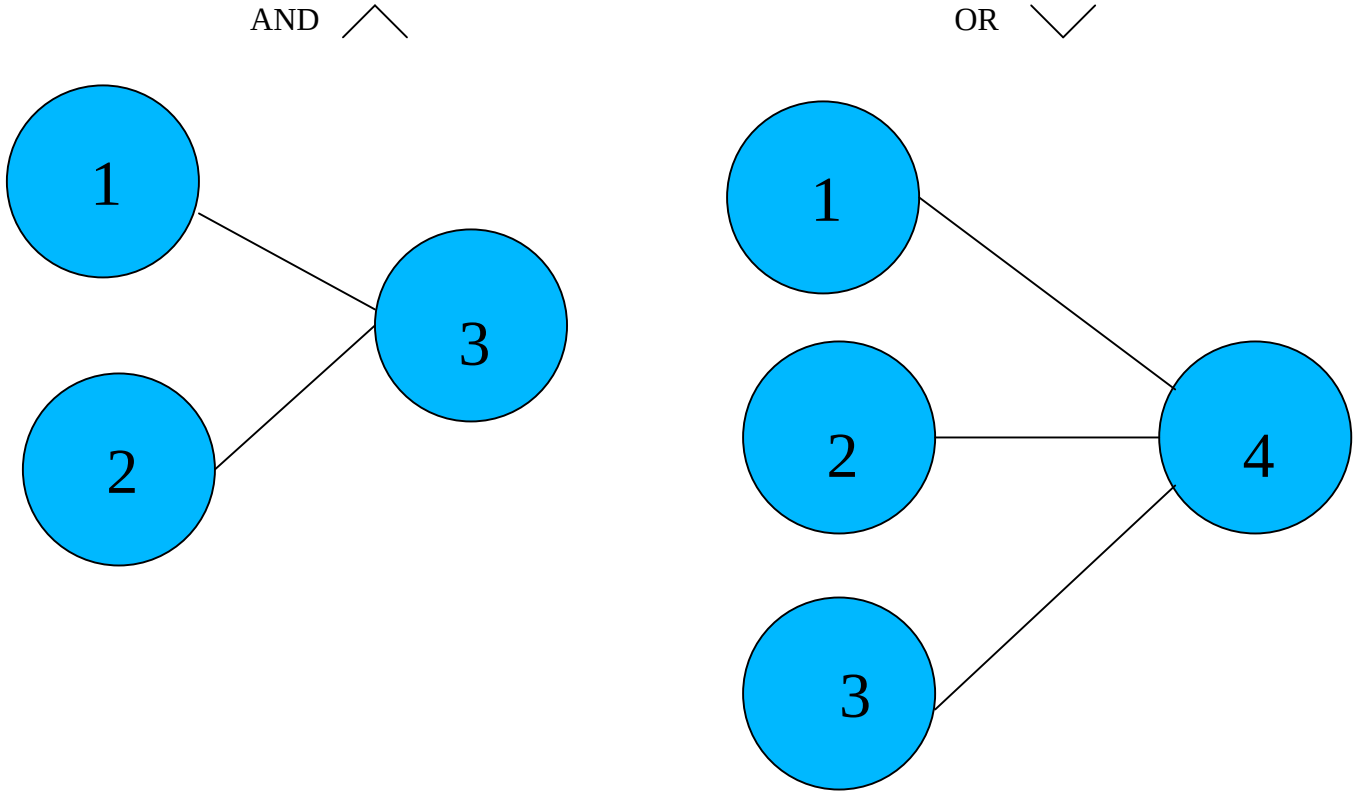
Ursache-Wirkungs-Graph

- Verbesserung von Äquivalenzklassen und Grenzwertanalyse
- eine formale Sprache, in die eine Spezifikation aus der natürlichen Sprache übersetzt wird
- Operatoren aus der booleschen Algebra
- Beispiel für die Notation eines Ursache-Wirkungs-Graphs :



Testen : Black-Box-Test (15)

Ursache-Wirkungs-Graph



Testen : Black-Box-Test (16)

Beispiel von Ursache-Wirkungs-Graph

- Angenommen dass, wir eine Spezifikation für eine Methode haben, die dem Benutzer erlaubt, eine Suche nach einem Buchstabe in einer vorhandenen Zeichenkette durchzuführen. Die Spezifikation besagt, dass der Benutzer die Länge von Zeichenkette (bis zu 80) und den zu suchenden Buchstabe bestimmen kann.

  Wenn der gewünschte Buchstabe in der Zeichenkette erscheint, wird seine Position berichtet. Wenn der gewünschte Buchstabe nicht in der Zeichenkette erscheint, wird eine Meldung „nicht gefunden“ ausgegeben. Wenn ein Index verwendet wird, der nicht im zulässigen Bereich liegt, wird eine Fehlermeldung „out of range“ ausgegeben.

Testen : Black-Box-Test (17)

Beispiel von Ursache-Wirkungs-Graph

Jetzt werden die Ursachen und die Wirkungen anhand der Spezifikation festgelegt

Die Ursachen sind :

C1 : Positive Ganzzahl von 1 bis 80

C2 : Der zu suchende Buchstabe is in der Zeichenkette

Die Wirkungen sind :

E1 : Die Ganzzahl ist out of range

E2 : Die Position des Buchstabes in der Zeichenkette

E3 : Der Buchstabe wird in der Zeichenkette nicht gefunden

Der Verhältnis (der semantische Inhalt) wird wie folgt beschrieben :

Wenn C1 und C2, dann E2.

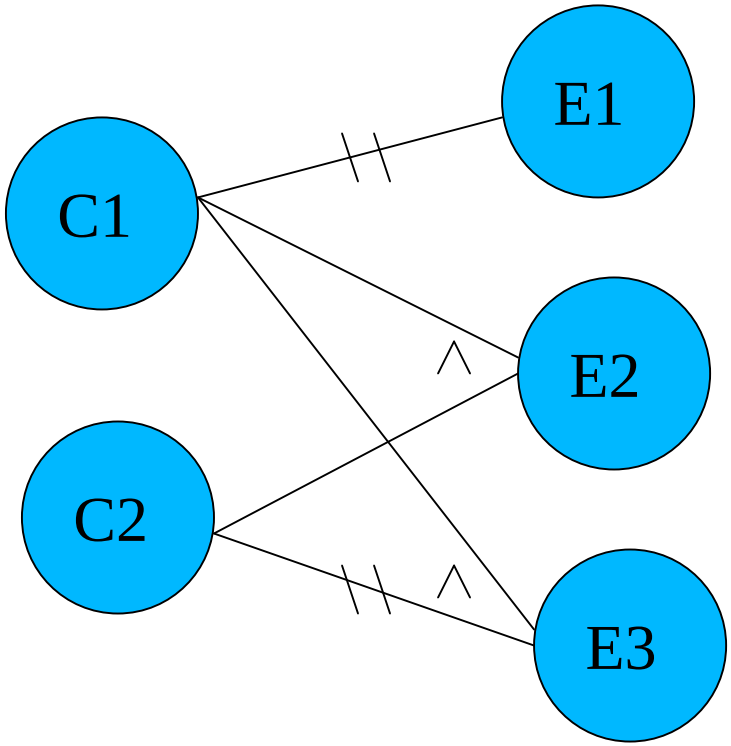
Wenn C1 und nicht C2, dann E3.

Wenn nicht C1, dann E1.

Testen : Black-Box-Test (18)

Beispiel von Ursache-Wirkungs-Graph

Im nächstens Schritt wird der Graph entwickelt Die Ursacheknoten werden auf einem Blatt Papier links und die Wirkungsknoten rechts notiert.



Testen : Black-Box-Test (19)

Beispiel von Ursache-Wirkungs-Graph

Der nächste Schritt ist Entscheidungstabelle zu bilden.

Testfall	T1	T2	T3
C1	1	1	0
C2	1	0	-
E1	0	0	1
E2	1	0	0
E3	0	1	0

T1, T2, T3 sind Testfälle
E1, E2, E3 sind die Wirkung (effect)
C1, C2 sind die Ursache (cause)
0 stellt den Zustand „nicht vorhanden“, „falsch“ dar
1 stellt den Zustand „vorhanden“, „wahr“ dar
- stellt den Zustand do not care

Testen : Black-Box-Test (20)

Beispiel von Ursache-Wirkungs-Graph

Der Tester kann die Entscheidungstabelle benutzen, um das Programm zu testen. Zum Beispiel : Wenn die vorhandene Zeichenkette „abcde“ ist, sind die möglichen Testfälle wie folgt :

Testfälle	Länge	Der zu suchende Buchstabe	Ausgabe
T1	5	c	3
T2	5	w	Nicht gefunden
T3	90	-	Out of range

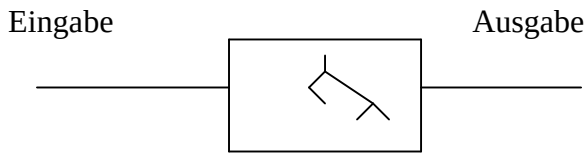
Testen : White-Box-Test (1)

- Was ist White-Box-Test ?
- Wichtiges Verfahren bei White-Box-Test : Beweise durch Widersprüche, Testdeckungsgrad
- Vorteile und Nachteile von White-Box-Test

Testen : White-Box-Test (2)

Was ist White-Box-Test ?

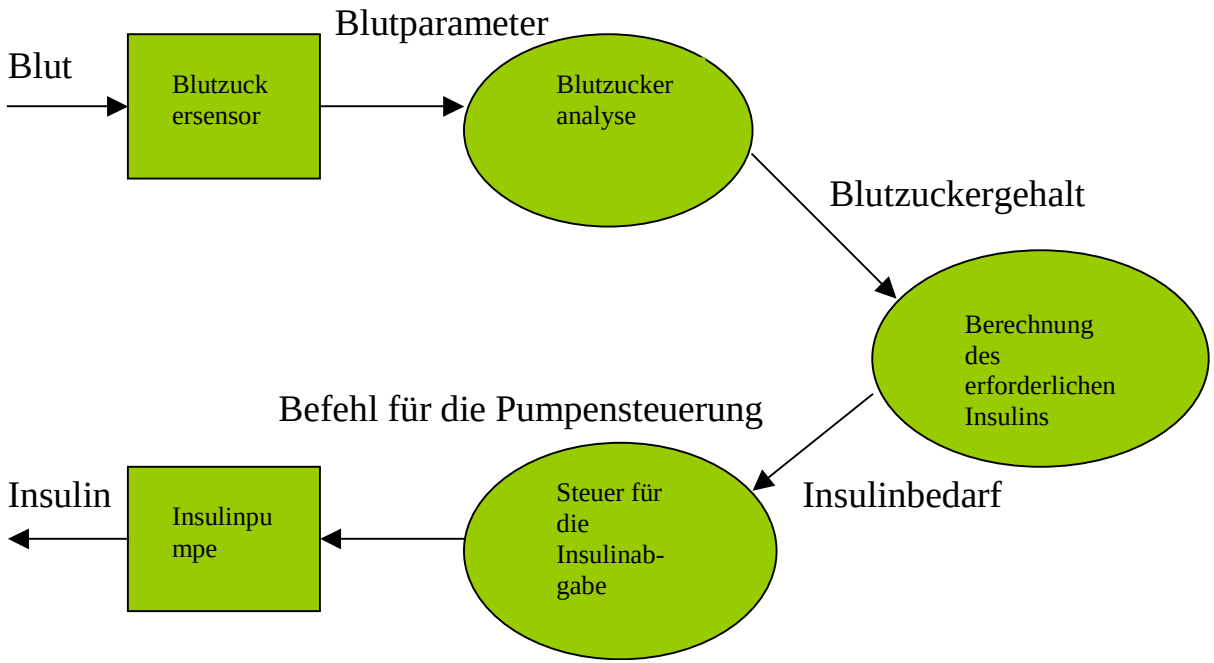
- Der Tester muss die innere Struktur des Programms und die Quellcode kennen
- Jeder Programmpfad muss mindestens einmal duchlaufen werden.
- Jeder Modul, jede Funktion muss mindestens einmal benutzt werden



Testen : White-Box-Test (3)

Beweise durch Widersprüche

Insulindosiersysteme



Testen : White-Box-Test (4)

Beweise durch Widersprüche

```
// Quellcode für die Insulinabgabe
// Die abgegebene Insulinmenge ist eine Funktion des Blutzuckerspiegels, der zuletz
// abgegebenen Dosis und der Zeit, zu der die letzte Dosis verabreicht wurde.

CurrentDose = computeInsulin ();

//Sicherheitsüberprüfung- Anpassung der aktuellen Dosis, wenn nötig
//if Anweisung 1

if (previousDose == 0)
{
      if (currentDose > 16)
          currentDose = 16;
}
else
     if (currentDose > (previousDose * 2) )
          currentDose = previousDose * 2;

//if Anweisung 2

if ( currentDose < minimumDose)
     currentDose = 0;
else
     if ( currentDose > maxDose)
           currentDose = maxDose;
administerInsulin (currentDose);
```

Testen : White-Box-Test (5)

Beweise durch Widersprüche

- Wenn wir den Code analysieren, gibt es drei mögliche Programmpfade, die zum Aufruf der Methode administerInsulin führen :

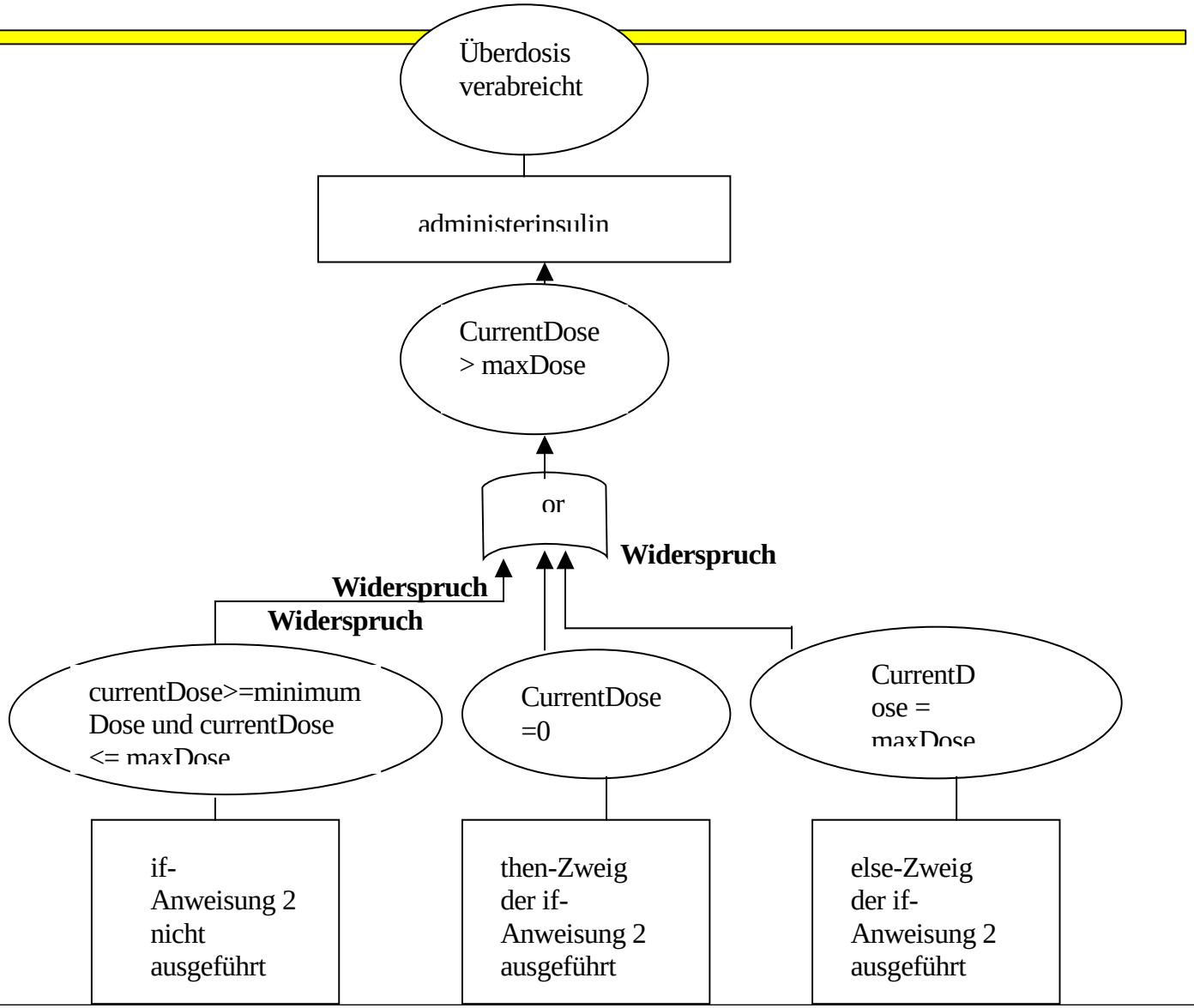
  1. Keiner der Zweige der if-Anweisung 2 wird ausgeführt. Das kann nur geschehen, wenn currentDose grösser oder gleich minimumDose und kleiner oder gleich maxDose ist.

  2. Der then-Zweig von if-Anweisung 2 wird ausgeführt. In diesem Fall wird currentDose auf Null gesetzt. Seine Nachbedingung lautet currentDose = 0.

  3. Der else-if Zweig der if-Anweisung 2 wird ausgeführt. In diesem Fall wird currentDose auf maxDose gesetzt. Die Nachbedingung lautet currentDose = maxDose.

Testen : White-Box-Test (6)

Beweise durch Widersprüche



Testprinzipien (1)

- Ein Programmierer sollte nie versuchen, sein eigenes Programm zu testen
- Testfälle müssen für ungültige und unerwartete ebenso wie für gültige und erwartete Eingabedaten definiert werden
- Die Ergebnisse von Tests müssen gründlich untersucht und analysiert werden
- Die Wahrscheinlichkeit, in einem bestimmten Segment des Programmcodes in der näheren Umgebung eines bereits bekannten Fehler weitere Fehler zu finden, ist überproportional hoch
- Zu jedem Test gehört die Definition des erwarteten Ergebnisses vor dem Beginn des Test
- Testen ist definiert als die Ausführung eines Programms mit der erklärten Absicht, Fehler zu finden
- Plannen Sie nie einen Test in der Annahme, dass keine Fehler gefunden werden

Literaturverzeichnis

- **Literaturverzeichnis**

- [1] Georg Erwin Thaller. *Verifikation und Validation*. Vieweg, 1994.
- [2] Ian Sommerville. *Software Engineering.* Addison-Wesley, 2000.
- [3] Glenford J.Myers. *Methodisches Testen von Programmen.Oldenbourg,1991.*
- [4] Neil Storey. *Safety-Critical Computer Systems*. Prentince Hall, 1996.

Literaturverzeichnis

[5] Edward Kit. *Sofware Testing in The Real World*. Addison-Wesley, 1995.

[6] Ilene Burnstein. Practical Software Testing. Springer, 2003.

[7] Helmurt Balzert. Lehrbuch Grundlagen der Informatik. Spektrum, 1999.

[8] Hauptseminar Prof. Huckle : Therac 25

http://www5.in.tum.de/lehre/seminar/semsoft/unterlagen_02/therac/ website

[9] Torsten Bresser. *Validieren und Verifikation (inkl. Testen, Model-Checking und Theorem Proving)*. Seminar, 2004.

[10] Friederike Nickl. *Qualitätssicherung und Testen.* sepis