Test und Verlässlichkeit Grosse Übung zu Foliensatz 6

Prof. G. Kemnitz

11. Juni 2021

Contents

Inhalt: Große Übungen zu Foliensatz 6

Contents

1 Statische Tests

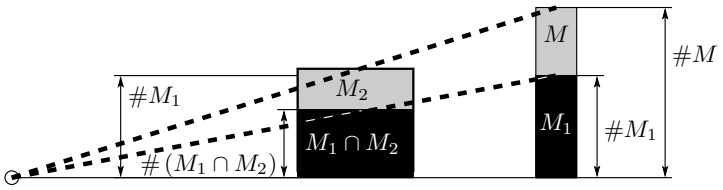
1.1 Inspektion

Aufgabe 6.1: Inspektionsfehlerüberdeckung

Inspektionsergebnisse für ein Programm aus 1000 Codezeilen:

- Inspekteur 1: 85 gefundene Fehler
- Inspekteur 2: 76 gefundene Fehler
- Schnittmenge: 56 übereinstimmende gefundene Fehler.

Schätzen Sie nach dem Verfahren »Capture-Recapture« die
a) Gesamtanzahl der Fehler?
b) Anzahl der nicht gefundenen Fehler?
c) Inspektionsfehlerüberdeckung?



a) Geschätzte Gesamtfehleranzahl:

$$\#F = \#M \approx \frac{\#M_1 \cdot \#M_2}{\#(M_1 \cap M_2)} = \frac{85 \cdot 76}{56} = 115{,}4$$

b) Geschätzte Anzahl der gefundenen Fehler:

$$\#EF \approx \#(M_1 \cup M_2) = 85 + 76 - 56 = 105$$

c) Inspektionsfehlerüberdeckung:

$$IFC \approx 1\frac{\#EF}{\#F} = \frac{105}{115{,}4} = 91\%$$

Aufgabe 6.2: Effizienz und Effektivität

In der Aufgabe zuvor hat der erste Inspekteur zehn Stunden für das Aufspüren seiner 85 gefundenen Fehler und der zweite Inspekteur 12 Stunden für das Aufspüren seiner 76 Fehler benötigt. Wie groß waren Effizienz[1] und Effektivität[2] beider Inspekteure einzeln und wie groß waren Effizienz und Effektivität der gesamten Inspektion?

Zur Kontrolle

	Insp. 1	Insp. 2	zusammen
gefundene Fehler	85	76	85+76-56=105
Zeit	10 h	12 h	22 h
Effizienz	8,5 $\frac{\text{Fehler}}{\text{h}}$	6,3 $\frac{\text{Fehler}}{\text{h}}$	4,8 $\frac{\text{Fehler}}{\text{h}}$
Effektivität	85 $\frac{\text{Fehler}}{1000\,\text{NLOC}}$	76 $\frac{\text{Fehler}}{1000\,\text{NLOC}}$	105 $\frac{\text{Fehler}}{1000\,\text{NLOC}}$

Aufgabe 6.3: Inspektion als Zufallstest

In einem Inspektionsprozess mit n Inspekteuren, die sich alle das System je 30 Stunden lang anschauen, betrage der Zusammenhang zwischen der Anzahl der nicht erkannten Fehler und der Anzahl der Inspekteure:

$$\#F(n) = \#F(1) \cdot \left(\frac{n}{1}\right)^{-k}$$

($\#F(1) = 100$ – zu erwartende Anzahl der nicht erkannte Fehler mit einem Inspekteur; $k = 0{,}5$ – Abnahmeexponent).

1. Wie viele Inspekteure sind erforderlich, um die zu erwartende Anzahl der nicht erkannten Fehler auf 25 zu reduzieren?
2. Bestimmen Sie die zu erwartende Effizienz für den zweiten bis fünften Inspekteur.

Zur Kontrolle

1. Anzahl der Inspekteure zur Reduzierung der zu erwartenden Anzahl der nicht nachweisbaren Fehler von 100 auf 25:

   $$n = \left(\frac{\#F(n)}{\#F(1)}\right)^{-\frac{1}{0{,}5}} = 4^2 = 16$$

   Zusätzlich zum ersten noch 15 weitere Inspekteure.

2. Zu erwartende Anzahl erkannter Fehler Inspekteur n:

   $$\begin{aligned} \#F(n) &= \#F(n-1) - \#F(n) \\ &= 100 \cdot \left(\frac{1}{\sqrt{n-1}} - \frac{1}{\sqrt{n}}\right) \end{aligned}$$

n	2	3	4	5
$\#F(n-1) - \#F(n)$	29,3	13,0	7,7	5,3
Effizienz* = $\frac{\#F(n-1)-\#F(n)}{30\,\text{h}}$	0,976	0,433	0,258	0,176

* in gefundenen Fehlern pro Stunde.

[1] Gefundene Fehler pro Mitarbeiterstunde.
[2] Gefundene Fehler auf 1000 Nettocodezeilen.

1.2 Typ, WB

Aufgabe 6.4: Typ und Wertebereichkontrollen

In VHDL seien folgende Typen und Variablen definiert:

```
type tWahrsch is range 0.0 to 1.0:
type tEX is 0.0 to 10.0;
variable w, w1, w2, w3, w4: tWahrsch;
variable EX: tEX;
```

Welche der nachfolgenden Zuweisungen sind

- typentechnisch erlaubt und
- welche weisen bei der Abarbeitung immer zulässige Werte zu?

```
n1: w3 := 0.5 + w1 * w2;
n2: w4 := (0.1*w1) + (0.9*w2);
n3: w := 1 - (1-w1)*(1-w2);
n4: Ex := 2.0 * (w1+w2+w3+w4);
```

Ergänzen Sie fehlende Typumwandlungen bei Typunverträglichkeit und Assert-Anweisungen vor möglichen Wertebereichsüberläufen.

Zur Kontrolle

```
type tWahrsch is range 0.0 to 1.0:
type tEX is 0.0 to 10.0;
variable w, w1, w2, w3, w4: tWahrsch;
variable EX: tEX;
...
n1: assert w1*w2<=0.5;
    w3 := 0.5 + w1 * w2;
n2: w4 := (0.1*w1) + (0.9*w2);
n3: w  := 1.0 - (1.0-w1)*(1.0-w2);
n4: Ex := 2.0 * (tEx(w1)+tEx(w2)+tEx(w3)+tEx(w4));
```

n1:	Assert-Anweisung ergänzt
n2:	alle Typ- und WB-Zuordnungen o.k.
n3:	ganzzahlige »1« durch »1.0« ersetzt
n4:	Konvertierungungen von »tWahrsch« nach »tEx«

1.4 Statische Code-Analyse

Aufgabe 6.5: Statische Code-Analyse

Nennen Sie drei Kontrollmöglichkeiten für Software, die der statischen Code-Anlyse zuzuordnen sind.

Zur Kontrolle

- Kontrolle, dass alle Variablen vor ihrerer ersten Nutzung initialisiert werden.
- Kontrolle der Einhaltung von API-Benutzerregeln durch Treiber.
- Kontrolle auf Nichtverwendung von Code-Bausteinen, die als problematisch gelten, z.B. in C »strcpy()«.

Aufgabe 6.6: C-typischer Multiplikationsfehler

Das Unterprogramm

```
uint32_t umult16(uint16_t a, uint16_t b){
 return a*b;
}
```

hat einen C-typischen Multiplikationsfehler. Für $1000 \cdot 1000 = 1000000$ berechnet es z.B. statt 1.000.000 nur 16.960.

1. Welchen Fehler hat das Programm?
2. Welche Regel für eine statische Code-Kontrolle lässt sich aus dem Beispiel ableiten?

Zur Kontrolle

1. Die Ursache der Fehlfunktion wird offensichtlich, wenn die Rechung hexadezimal erfolgt:
   - Sollergebnis: 0xf4240
   - Ist-Ergebnis: 0x4240

   Die führenden 2 Byte werden auf 0 gesetzt, weil bei C ein 16×16-Bit-Produkt nur 16 Bit groß ist. Der Cast auf 32 Bit erfolgt erst danach. Um ein 32-Bit-Produkt zu erhalten, muss mindestens ein Summand vor der Multiplikation auf 32 Bit gecastet werden:

   ```
   uint32_t umult16(uint16_t a, uint16_t b){
    return (uint32_t)a*b;
   }
   ```

2. Regel: Kontrolliere für jedes ganzzahlige Produkt, dass Ist- und Soll-Ergebnistyp übereinstimmen.

2 Testauswahl

2.3 Def-Use-Ketten

Aufgabe 6.7: Def-Use-Ketten

```
int ggt(int a, int b){
n0:   int c = a;
n1:   int d = b;
n2:   if(c == 0)
n3:     return d;
n4:   while(d != 0){
n5:     if(c > d)
n6:       c = c - d;
n7:     else
n8:   }   d = d - c;
n9: } return c;
```

1. Ergebnis von »n6« sei verfälscht. Möglichen »Defs«?
2. Wie vereinfacht sich die Rückverfolgung von Verfälschungen bei Aufzeichnung aller Anweisungsergebnisse als Trace?

Lösung

```
int ggt(int a, int b){
n0:   int c = a;
n1:   int d = b;
n2:   if(c == 0)
n3:     return d;
n4:   while(d != 0){
n5:     if(c > d)
n6:       c = c - d;
n7:     else
n8:   }   d = d - c;
n9: } return c;
```

1. mögliche »Defs« für die Variable c: »n0« und »n6«. Mögliche »Defs« für die Variable d: »n1« und »n8«.
2. Erspart bei Rückverfolgungsschritten die Testwiederholung bis zu den potentiellen »Defs« davor.

2.4 Äquivalenzklassen

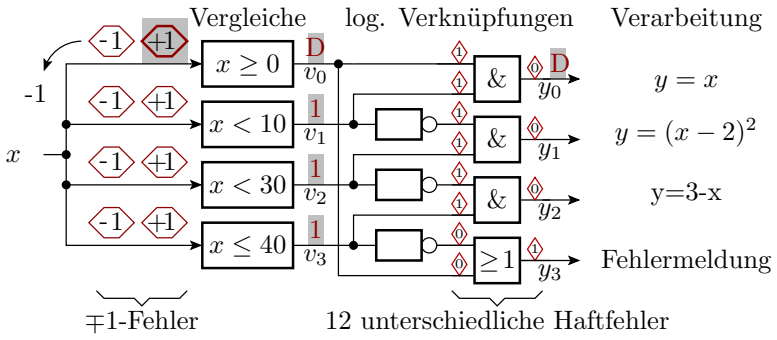
Aufgabe 6.8: Äquivalenzklassen

Gegeben ist die als Tabelle spezifizierte Funktion:

x	y
$0 \leq x < 10$	$y := x$
$10 \leq x < 30$	$y := (x-2)^2$
$30 \leq x \leq 40$	$y := 3 - x$
sonst	Fehlermeldung

1. Skizzieren Sie den Berechnungsfluss für eine äquivalenzklassenbasierte Testauswahl.
2. Zeichnen Sie alle nicht äquivalenten ∓ 1[3]- und sa-Fehler ein.
3. Berechnen Sie einen Test für den +1-Fehler der Bedingung $(0 \leq x)$.

Lösung



Der +1-Fehler verlangt zur Anregung $x = -1$ und ist an

- $y_1 = \mathrm{D}$ bzw.
- »y=x« wird nicht ausgeführt

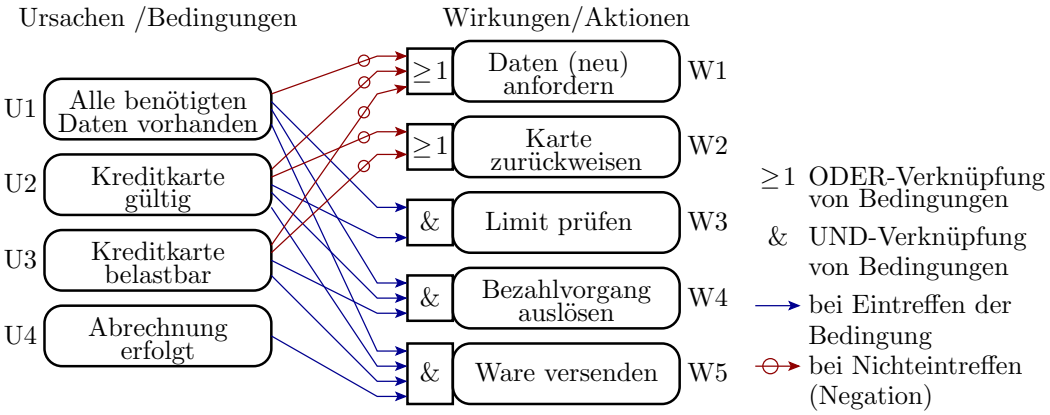
beobachtbar.

[3] Off-by-One-Fehler.

2.5 UW-Analyse

Aufgabe 6.9: Ursache-Wirkungs-Analyse

Gegeben ist das Ergebnis einer Ursache-Wirkungs-Analyse in einer anderen Darstellung aus [http://test.silke-wingens.de/].



1. Stellen Sie die dargestellte Ursache-Wirkungs-Beziehung als logischen Signalflussplan dar.

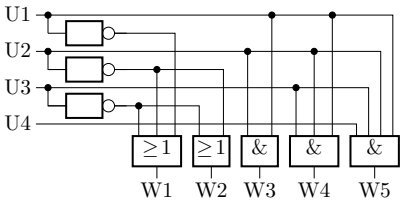
2. Bestimmen Sie in dieser Darstellung für Wirkung W3 die Menge der unterschiedlich nachweisbaren Haftfehler ohne redundante und implizit nachweisbare Fehler.

3. Suchen Sie für alle (drei) Haftfehler eine Menge von Ursachenkombinationen, mit denen sie anhand ihrer Wirkung nachweisbar sind.

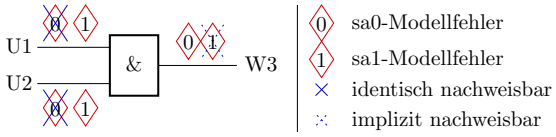
4. Bestimmen Sie für die (drei) Haftfehler die Nachweiswahrscheinlichkeiten für die Auftrittshäufigkeiten der Ursachen $h\,(\mathrm{U1}) = 30\%$, $h\,(\mathrm{U2}) = 70\%$, $h\,(U3) = 20\%$ und $h\,(U4) = 80\%$.

Lösung Aufgabenteil 1 und 2

1. Ursache-Wirkungs-Beziehung als logischen Signalflussplan:



2. Anfangsfehlermenge 6 Haftfehler. sa0(U1), sa0(U2) und sa0(W3) sind identisch und sa1(W3) implizit von sa1(U1) und sa1(U2) nachweisbar:



Lösung Aufgabenteil 3 und 4

3. Möglicher Testsatz:

Fehler:	sa1(U1)	sa1(U2)	sa0(W2)
Test:	U1=0, U2=1	U1=1, U2=0	U1=1, U2=1

4. Nachweiswahrscheinlichkeit für $h\,(\mathrm{U1}) = 30\%$ und $h\,(\mathrm{U2}) = 70\%$:

U2	U1	Auftrittshäufigkeit	sa1(U1)	sa1(U2)	sa0(W2)
0	0	$30\% \cdot 70\% = 21\%$			
0	1	$30\% \cdot 30\% = 9\%$		x	
1	0	$70\% \cdot 70\% = 49\%$	x		
1	1	$70\% \cdot 30\% = 21\%$			x
Nachweiswahrscheinlichkeit:			49%	9%	21%