Test und Verlässlichkeit
Grosse Übung zu Foliensatz 4

Prof. G. Kemnitz

Institut für Informatik, TU Clausthal (TV_GUeF4)

28. Juni 2018

Inspektion

Aufgabe 4.1: Inspektionsfehlerüberdeckung

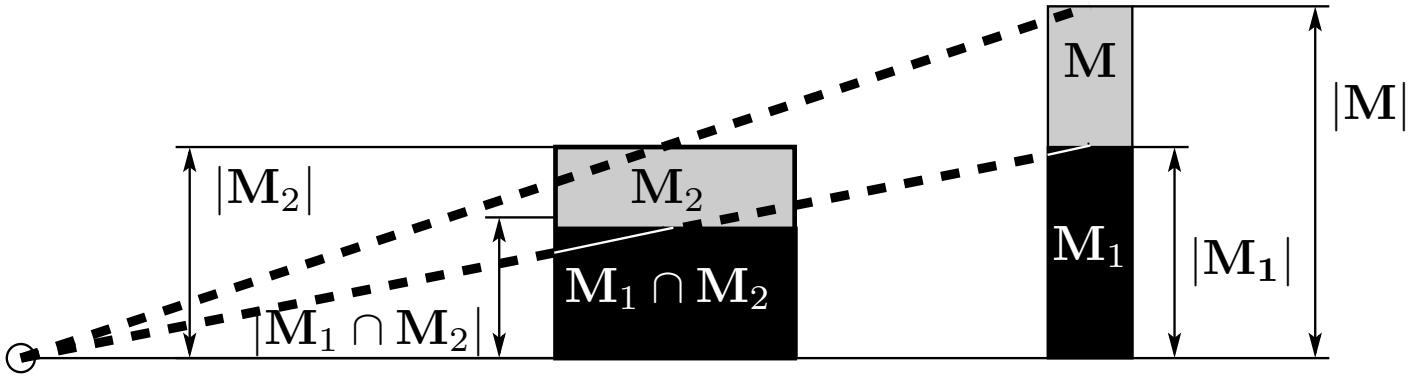
Inspektionsergebnisse für ein Programm aus 1000 Codezeilen:

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

Schätzen Sie nach dem Verfahren »Capture-Recapture«

1. die Gesamtanzahl der Fehler,
2. die Anzahl der nicht gefundenen Fehler und
3. die Inspektionsfehlerüberdeckung.

Zur Kontrolle



1. Geschätzte Gesamtfehleranzahl:

$$\varphi = |\mathbf{M}| \approx \frac{|\mathbf{M}_1| \cdot |\mathbf{M}_2|}{|\mathbf{M}_1 \cap \mathbf{M}_2|} = \frac{85 \cdot 76}{56} = 115{,}4$$

2. Geschätzte Anzahl der nicht gefundenen Fehler:

$$\varphi_{\mathrm{NErk}} \approx |\mathbf{M}| - |\mathbf{M}_1 \cup \mathbf{M}_2| = 115{,}4 - (85 + 76 - 56) = 10{,}4$$

3. Inspektionsfehlerüberdeckung:

$$IFC \approx 1 - \frac{\varphi_{\mathrm{NErk}}}{\varphi} = 1 - \frac{10{,}4}{115{,}4} = 91\%$$

Aufgabe 4.2: 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?

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

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 4.3: 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:

$$E\left(\varphi_{\mathrm{NErk}}\left(n\right)\right) = E\left(\varphi_{\mathrm{NErk}}\left(1\right)\right) \cdot \left(\frac{n}{1}\right)^{-k}$$

($E\left(\varphi_{\mathrm{NErk}}\left(1\right)\right) = 100$ – 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:

$$n = \left(\frac{E\left(\varphi_{\mathrm{NErk}}\left(n\right)\right)}{E\left(\varphi_{\mathrm{NErk}}\left(1\right)\right)} \right)^{-\frac{1}{0{,}5}} = 4^2 = 16$$

   Zusätzlich zum ersten noch 15 weitere Inspekteure.
2. Zu erwartende Anzahl erkannter Fehler Inspekteur n:

$$\begin{aligned} E\left(\varphi_{\mathrm{Erk}}\left(n\right)\right) &= E\left(\varphi_{\mathrm{NErk}}\left(n-1\right)\right) - E\left(\varphi_{\mathrm{NErk}}\left(n\right)\right) \\ &= 100 \cdot \left(\frac{1}{\sqrt{n-1}} - \frac{1}{\sqrt{n}} \right) \end{aligned}$$

n	2	3	4	5
$E\left(\varphi_{\mathrm{Erk}}\left(n\right)\right)$	29,3	13,0	7,7	5,3
Effizienz* $\frac{E(\varphi_{\mathrm{Erk}}(n))}{30\,\mathrm{h}}$	0,976	0,433	0,258	0,176

(Zu erwartende Effizienz in gefundenen Fehlern pro Stunde).

Statische Typ- und WB-Kontrollen

Aufgabe 4.4: 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«

Statische Code-Analyse

Aufgabe 4.5: 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 4.6: 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.

Baugruppen

Aufgabe 4.7: MDA und ICT

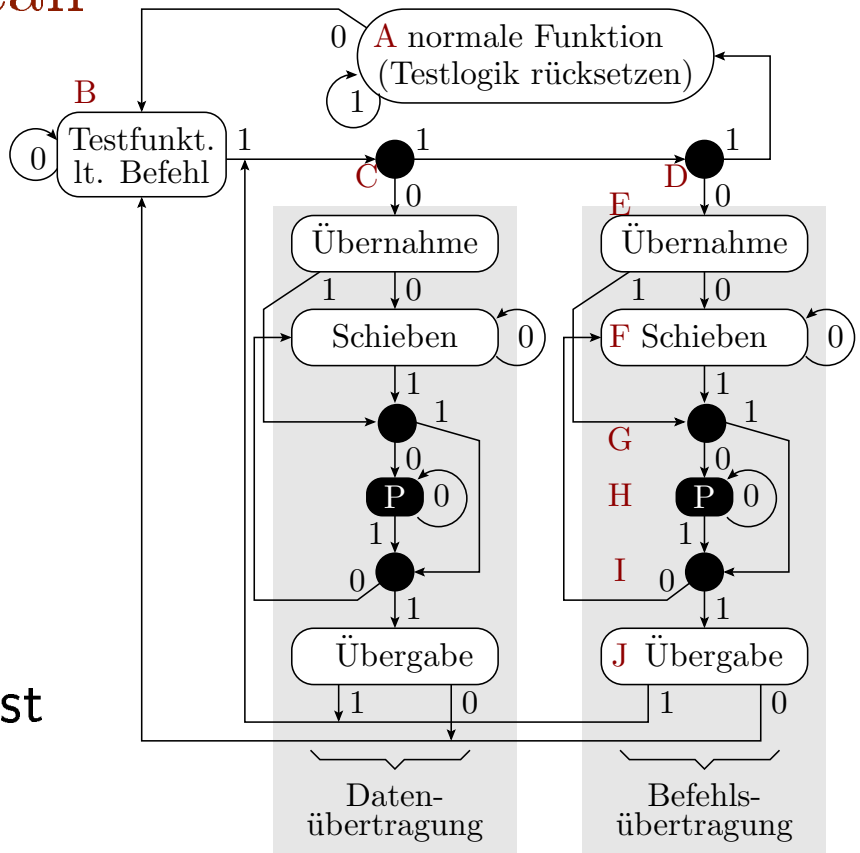
1. Wodurch unterscheidet sich der analoge In-Circuit-Test von einer Zweipunktmessung zur Kontrolle auf Fertigungsfehler?
2. Ersetzt ein digitaler In-Circuit-Test Zweipunktmessungen zur Kontrolle auf Fertigungsfehler vollständig?
3. Was bedeutet bei Boundry-Scan »Ersatz der Nadelbettadapters durch Silicon Nails«?

Zur Kontrolle

1. Die Strom-Spannungs-Beziehung einer Zweipunktmessung zur Kontrolle auf Fertigungsfehler hängt außer vom Bauteil zwischen den Punkten auch von der umgebenden Schaltung ab. Beim analogen In-Circuit-Test werden die wegfließenden Ströme von einem der Punkte unterdrückt und so die Abhängigkeit der gemessenen Strom-Spannungs-Beziehung von anderen Bauteilen unterbunden. Vereinfacht die Testerstellung.
2. Kein vollständiger Ersatz. Digitaler ICT ist ein Test unter Spannung. Mindestens zur Kontrolle und Beseitigung von Kurzschlüssen vor Anlegen der Spannung sind Zweipunktmessung zwischen den Leitungen erforderlich.
3. Bei Boundary-Scan erfolgt der Lese- und Schreibzugriff der logischen Pegel der Leitungen auf einer Baugruppe mit integrierten Teststrukturen (Schieberegisterringen) statt der Kontaktierung mit einem Nadeladapter.

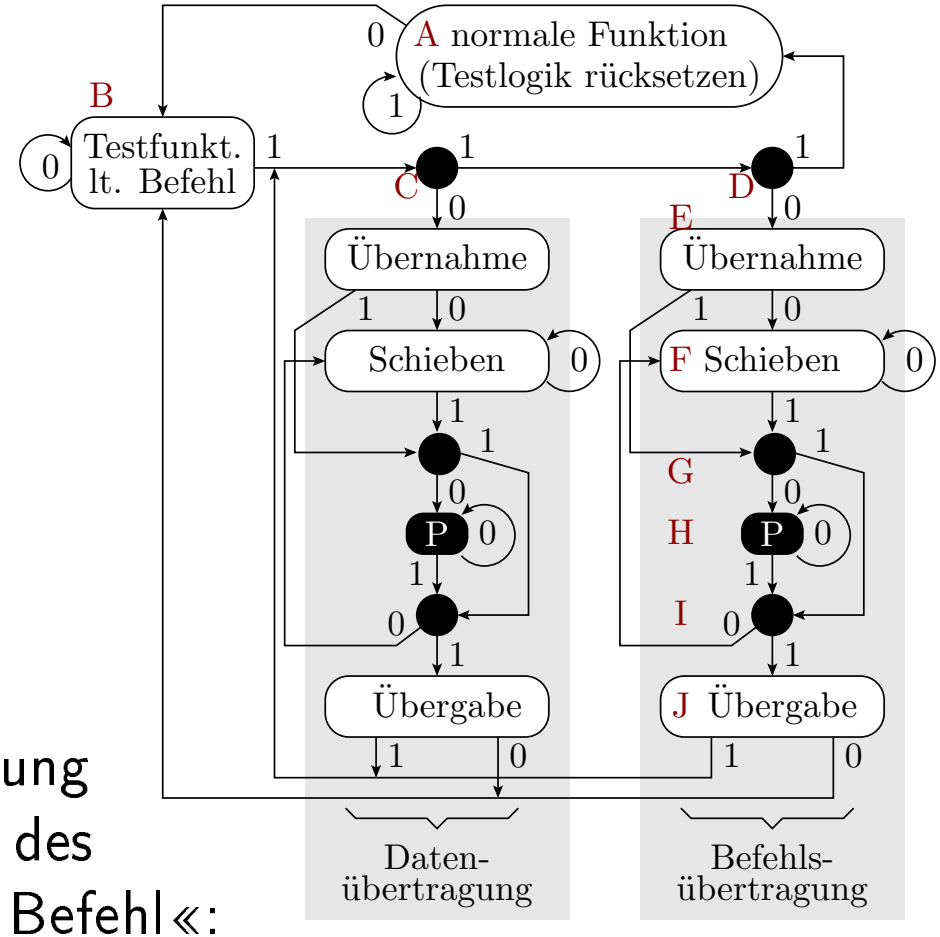
Aufgabe 4.8: Boundary-Scan

1. Mit welcher Bitfolge an TMS lässt sich der TAP-Controller aus einem beliebigen Zustand in den Zustand »normale Funktion« versetzen?
2. Mit welcher Bitfolge an TMS und TDI lässt sich ausgehend vom Zustand »normale Funktion« das 8-Bit-Befehlswort 0x4D und anschließend der Zustand »Testfunktion laut Befehl« einstellen?



Zur Kontrolle

1. TMS-Folge, um den TAP-Controller aus einem beliebigen Zustand in den Zustand »normale Funktion« zu versetzen: 5-mal '1'.
2. Zustands- und Bitfolge an TMS und TDI zur Einstellung des Befehlswort 0x4D und des Zustand »Testfunktion lt. Befehl«:



Takt	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Zustand	A	B	C	D	E	F	F	F	F	F	F	F	F	G	I	J	B
TMS	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	
TDI	–	–	–	–	–	1	0	1	1	0	0	1	0	–	–	–	