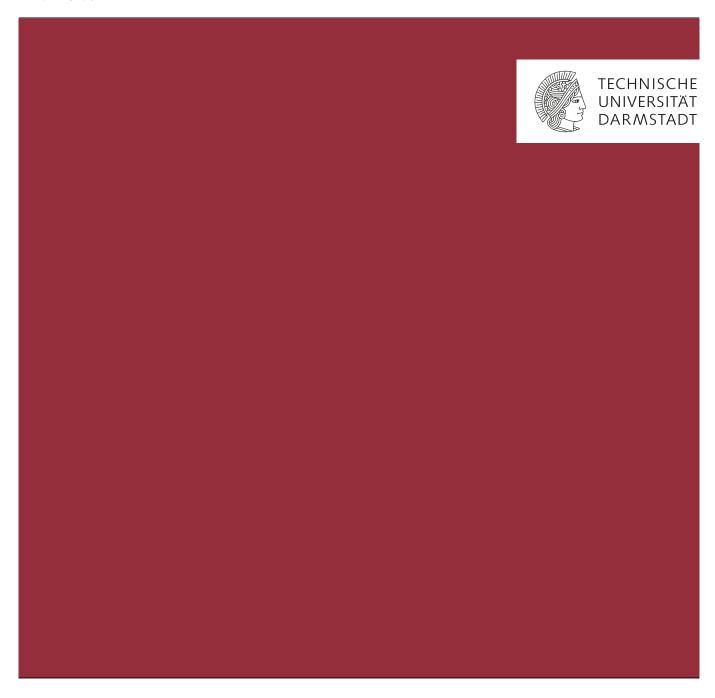
Software Engineering Übung 08

Verifikation

Übung von Jonathan Lippert und Magnus Dierking Tag der Einreichung: 22. Januar 2021

Darmstadt



Software Engineering Übung 08 Verifikation

Übung von Jonathan Lippert und Magnus Dierking

Tag der Einreichung: 22. Januar 2021

Darmstadt

1 Systematisches Testen von Methoden

1.1 Branch-Coverage

Testeingabe	Erwartetes Ergebnis/Exception		
hexDigitSum(null)	NullPointerException		
hexDigitSum(G)	-1		
hexDigitSum(3)	3		
hexDigitSum(B)	11		

1.2 Condition-Coverage

Testeingabe	Erwartetes Ergebnis/Exception			
hexDigitSum(5)	5			
hexDigitSum(B)	11			
hexDigitSum(G)	-1			
<pre>hexDigitSum(=)</pre>	-1			
hexDigitSum(%)	-1			

2 MCDC Testabdeckung

2.1 a)

Testeingabe	Erwartetes Ergebnis/Exception	c1	c2	c3	c4	Decision
hexaDigitSum(?)	false	false	true	true	false	false
hexaDigitSum(7)	true	false	false	true	false	true

2.2 b)

Listing 2.1: Conditions der aufgabe 2b

```
if(
currNum<0 | // Condition c1
(currNum>=0 & // Condition c2
((currNum>9 & // Condition c3
currNum<'A' // Condition c4
) |
currNum>'F')) // Condition c5
```

Angenommen wir wollen die MCDC für c = c3 anwenden. Da die Decision einmal Wahr und einmal Falsch sein soll, aber die restlichen Conditions gleich bleiben sollen, müssen die Oder-Verknüpften Conditions Falsch und die Und-Verknüpften Wahr ergeben. Also muss C1 = Wahr und C2 = Falsch gelten. Dies ist jedoch ein Widerspruch, da sie einander disjunkte Ereignisse darstellen. Mit Hilfe des Widerspruches ist also bewiesen, dass dies nicht für alle Conditions möglich ist.