Software Engineering – V-Aufgaben 8 - Gruppe A4
Jeff Wagner : 544167

Aufgabe 1 – Software-Metrik

-> Was ist eine Software-Metrik? Welche Metriken lassen sich den Phasen des Software-Entwicklungsprozesses Zuordnen ? Des Weiteren 3 Code- und OO-Metriken erläutern….

Eine Softwaremetrik, oder kurz Metrik, ist eine (meist mathematische) Funktion,
die eine Eigenschaft von Software in einen Zahlenwert, auch Maßzahl genannt,
abbildet. Hierdurch werden formale Vergleichs- und Bewertungsmöglichkeiten geschaffen.

3 Code-Metriken: *LOC, Halstead, McCabe*
3 OO-Metriken: *Grad der Objektorientiertheit, Bindung, Kopplung*

Beschreibungen aller Metriken sind hier zu finden:

--> http://www.ndepend.com/Metrics.aspx

WMC - Weighted Methods for Class
NOC - Number of Children
DIT - Depth of Inheritance Tree

> WMC - Doesn't really do what it claims to do - replace with either number of methods or total cyclomatic complexity
> NOC - A high level may be either a good or a bad thing, potential for confusion in measurement, particularly with regard to interfaces
> DIT - In general, but not always, a high DIT is viewed as a good thing - so you can use this as an indicator and then use your judgement after a visual examination of the class. There is some debate about calculation and, in the case of Java, what do we do about interfaces?

WMC, DIT und NOC sind zum Identifizieren der Klassen gedacht.
WMC überprüft zudem die Semantik einer Klasse.

--> http://www.virtualmachinery.com/sidebar3.htm

Aufgabe 2 – Halstead-Metriken („*complexity6*“)

-> Berechnen Sie für das Beispiel „complexity6“ die Halstead-Metriken n, N und D…

Halstead-Metriken am Bsp. „complexity6(int i, int j)”

Anzahl unterschiedlicher Operatoren n1: 11 *(void,int,if,while,>,==,--,%,(),{},&&)*
Anzahl unterschiedlicher Operanden n2: 5 *(i,j,0,1,2)*
n = n1 + n2 = 16

Gesamtzahl der verwendeten Operatoren N1: 26
Gesamtzahl der verwendeten Operanden N2: 17
N = N1 + N2 = 43

D = (n1 * N2) / (2 * n2) = (11 * 17) / (2 * 17) = 5,5

Aufgabe 3 – Zyklomatische Komplexität

-> Erläutern Sie die verschiedenen Verfahren und geben Sie die zyklomatische Komplexität der Kontrollflussgraphen In Abbildung 1 und des Übungsbeispiels „complexity6“…

McCabe-Metrik (zyklomatische Komplexität)
Methode: Binäre Prädikate zählen (s. Folien McCabe / Bothe)
Formel: binäre Prädikate p + 1.

complexity6:

1) 6

Abb. 1.:

1) 2
2) 2
3) 2
4) 2
5) 2
6) 1
7) 4

--> Das McCabe-Tool führt unter Umständen zu anderen Ergebnissen, da es Knoten u. U. zusammenfasst.

Aufgabe 4 – Kohäsion

-> Definieren Sie den Begriff Kohäsion und geben Sie 2 Maße an, um die Kohäsion einer Software-Komponente zu messen…

Die Kohäsion beschreibt die logische Beziehung zwischen Elementen (Daten, Operationen) innerhalb einer Komponente.

2 Maße zum Messen:
> Anzahl von Attributen einer Klasse
> Anzahl von Methoden einer Klasse

7 Methoden
7 Attribute

Aufgabe Z5 auf Folgeseite…

Aufgabe Z5 – Zyklomatische Komplexität (SOTA)

-> Nutzen Sie SOTA um die zyklomatische Komplexität der der Methoden „Error.error(…)“ und „Scanner.getSymbol(…)“ zu bestimmen und geben Sie diese an…

Name	Cycl.Compl.	Ess.Compl.	LOC	#Statements	#Branches	#ModBIP	#BIP	#ConditionStmts.	#Atoms	#Conditions
	< 30	< 22	581	396	153	818	3347	51	64	80
Error.java	< 30	< 2	59	91	29	31	31	---	---	---
Error	< 30	< 2	52	91	29	31	31	---	---	---
error (int,int,int)	29	1	39	88	29	29	29	---	---	---
symbolError (int,int,int	1	1	3	2	---	1	1	---	---	---
ioError ()	1	1	2	1	---	1	1	---	---	---
Scanner.java	< 28	< 22	346	219	114	752	3261	46	54	64
Scanner	< 28	< 22	336	219	114	752	3261	46	54	64
Scanner (InputStreamF	2	1	12	9	2	2	2	---	---	---
charIsDigit (char)	2	1	7	3	2	2	2	1	2	3
charIsAlpha (char)	2	1	7	3	2	2	2	1	4	7
readNextChar ()	2	1	10	7	2	2	2	---	---	---
getSymbol ()	27	21	78	89	36	653	757	9	10	11
handleIdentifier (String	20	20	21	39	38	20	20	19	19	19
number ()	12	1	70	50	22	53	2457	11	14	19
comment ()	4	1	15	9	6	13	13	3	3	3
getLine ()	1	1	2	1	---	1	1	---	---	---
getPos ()	1	1	2	1	---	1	1	---	---	---
main (String)	3	1	16	8	4	3	4	2	2	2