Sri Lanka Institute of Information Technology

SOFTWARE ENGINEERING PROCESS AND QUALITY MANAGEMENT

Lecture 7 – Software Metrics (Weighted Composite Complexity Metric)

Weighted Composite Complexity (WCC)

- Measure the complexity of the program.
- Object Oriented Metric
- Based on 4 key factors
  - Size
  - Type of control structures
  - Nesting level of control structures
  - Inheritance level of statements

Computing the WCC Value

- WCCM value of a program = $\sum_{j=1}^{n} S_j * (W_t)_j$
  - Sj = Size of j^{th} executable statement in terms of token count
  - n = Total number of executable statements in a program
  - (Wt)j = Total weight of the j^{th} executable statement in the program

- Wt = Wc + Wn + Wi
  - Wc = Weight due to type of control structures
  - Wn = Weight due to nesting level of control structures
  - Wi = Weight due to inheritance level of statements

Identify the Size of a Statement

The **Size (S)** of a statement is the **total number of tokens** it contains.

In WCC, a **token** is a fundamental program element used to measure the **size** of a statement.

However, **not everything in the code is a token**.

Refer to the guidelines document to identify the tokens in a program statement. (uploaded in Courseweb)

Identify the tokens and the size value of each statement of the following program

Line No	Program Statements
1	public class Result{
2	public void outresult(int marks) {
3	if (marks > -1 && marks < 50)
4	System.out.println("Fail");
5	else
6	System.out.println("Pass");
	}
7	public static void main(String args[ ]){
8	Result r = new Result( );
9	r.outresult(50);
	}
	}

Line No	Program Statements	Tokens	Size (S)
1	public class Result{		
2	public void outresult (int marks) {	void, outresult( )	2
3	if (marks > -1 && marks < 50)	if-else( ), marks, >, -1, &&, marks, <, 50	8
4	System.out.println("Fail");	System, ., out, ., println( ), "Fail"	6
5	else		
6	System.out.println("Pass");	System, ., out, ., println( ), "Pass"	6
	}		
7	public static void main(String args[ ]){	void, main( )	2
8	Result r = new Result( );	Result, r, =, new, Result( )	5
9	r.outresult(50);	r, ., outresult( )	3
	}		
	}		

Weight due to Type of Control Structure (Wc)

Type of control structure	Weight
Sequential	0
Branch	1
Iterative	2
Switch statement with **n** cases	n

Line No	Program Statements	Tokens	S	Wc
1	public class Result{			
2	public void outresult (int marks) {	void, outresult( )	2	0
3	if (marks > -1 && marks < 50)	if-else( ), marks, >, -1, &&, marks, <, 50	8	1
4	System.out.println("Fail");	System, ·, out, ·, println( ), "Fail"	6	0
5	else			
6	System.out.println("Pass");	System, ·, out, ·, println( ), "Pass"	6	0
	}			
7	public static void main(String args[ ]){	void, main( )	2	0
8	Result r = new Result( );	Result, r, =, new, Result( )	5	0
9	r. outresult(50);	r, ·, outresult( )	3	0
	}			
	}			

Weight due to Nesting level of Control Structure (Wn)

Nesting Level of Statements	Weight
Sequential statements	0
Statements inside the outer most level/first level of control structures	1
Statements inside the second level control structures	2
Statements inside the third level control structures	3
Statements inside the n^{th} level control structures	n

Line No	Program Statements	Tokens	S	Wc	Wn
1	public class Result{				
2	public void outresult (int marks) {	void, outresult( )	2	0	0
3	if (marks > -1 && marks < 50)	if-else( ), marks, >, -1, &&, marks, <, 50	8	1	1
4	System.out.println("Fail");	System, ·, out, ·, println( ), "Fail"	6	0	1
5	else				
6	System.out.println("Pass");	System, ·, out, ·, println( ), "Pass"	6	0	1
	}				
7	public static void main(String args[ ]){	void, main( )	2	0	0
8	Result r = new Result( );	Result, r, =, new, Result( )	5	0	0
9	r. outresult(50);	r, ·, outresult( )	3	0	0
	}				
	}				

Weight due to Inheritance level of Statements (Wi)

Inheritance Level of Statements	Weight
Statements inside the base class/root class	0
Statements inside the first derived class	1
Statements inside the second derived class	2
Statements inside the n^{th} derived class	n

Line No	Program Statements	Tokens	S	Wc	Wn	Wi
1	public class Result{					
2	public void outresult (int marks) {	void, outresult( )	2	0	0	1
3	if (marks > -1 && marks < 50)	if-else( ), marks, >, -1, &&, marks, <, 50	8	1	1	1
4	System.out.println("Fail");	System, ·, out, ·, println( ), "Fail"	6	0	1	1
5	else					
6	System.out.println("Pass");	System, ·, out, ·, println( ), "Pass"	6	0	1	1
	}					
7	public static void main(String args[ ]){	void, main( )	2	0	0	1
8	Result r = new Result( );	Result, r, =, new, Result( )	5	0	0	1
9	r. outresult(50);	r, ·, outresult( )	3	0	0	1
	}					
	}					

Total Weight (Wt)

Total Weight of a statement. It combines three dimensions of complexity into one value:

$$Wt = Wc + Wn + Wi$$

Wc = Weight due to Type of Control Structure
Wn = Weight due to Nesting Level
Wi = Weight due to Inheritance Level

Line No	Program Statements	Tokens	S	Wc	Wn	Wi	Wt
1	public class Result{						
2	public void outresult (int marks) {	void, outresult( )	2	0	0	1	1
3	if (marks > -1 && marks < 50)	if-else( ), marks, >, -1, &&, marks, <, 50	8	1	1	1	3
4	System.out.println("Fail");	System, ·, out, ·, println( ), "Fail"	6	0	1	1	2
5	else						
6	System.out.println("Pass");	System, ·, out, ·, println( ), "Pass"	6	0	1	1	2
	}						
7	public static void main(String args[ ]){	void, main( )	2	0	0	1	1
8	Result r = new Result( );	Result, r, =, new, Result( )	5	0	0	1	1
9	r. outresult(50);	r, ·, outresult( )	3	0	0	1	1
	}						
	}						

Weighted Complexity for a Single Statement (WC)

It tells us how complex that one line is by factoring in:

- How many tokens it contains (Size S)
- How "heavy" or complex its context is (Total Weight Wt)

$$WC = S \times Wt$$

Line No	Program Statements	Tokens	S	Wc	Wn	Wi	Wt	WC
1	public class Result{							
2	public void outresult (int marks) {	void, outresult( )	2	0	0	1	1	2
3	if (marks > -1 && marks < 50)	if-else( ), marks, >, -1, &&, marks, <, 50	8	1	1	1	3	24
4	System.out.println("Fail");	System, ·, out, ·, println( ), "Fail"	6	0	1	1	2	12
5	else							
6	System.out.println("Pass");	System, ·, out, ·, println( ), "Pass"	6	0	1	1	2	12
	}							
7	public static void main(String args[ ]){	void, main( )	2	0	0	1	1	2
8	Result r = new Result( );	Result, r, =, new, Result( )	5	0	0	1	1	5
9	r. outresult(50);	r, ·, outresult( )	3	0	0	1	1	3
	}							
	}							

Weighted Composite Complexity for the Program (WCC)

The final WCC is the sum of all WC values for each line.

Line No	Program Statements	Tokens	S	Wc	Wn	Wi	Wt	WC
1	public class Result{							
2	public void outresult (int marks) {	void, outresult( )	2	0	0	1	1	2
3	if (marks > -1 && marks < 50)	if-else( ), marks, >, -1, &&, marks, <, 50	8	1	1	1	3	24
4	System.out.println("Fail");	System, ·, out, ·, println( ), "Fail"	6	0	1	1	2	12
5	else							
6	System.out.println("Pass");	System, ·, out, ·, println( ), "Pass"	6	0	1	1	2	12
	}							
7	public static void main(String args[ ]){	void, main( )	2	0	0	1	1	2
8	Result r = new Result( );	Result, r, =, new, Result( )	5	0	0	1	1	5
9	r. outresult(50);	r, ·, outresult( )	3	0	0	1	1	3
	}							
	}							
		WCC Value						**60**

THANK YOU!