Sri Lanka Institute of Information Technology

SOFTWARE ENGINEERING PROCESS AND QUALITY MANAGEMENT

Lecture 5 – Software Metrics (Cognitive Functional Size Metric)

Cognitive Functional Size Metric

- Paradigm Independent metric.

- CFS is a function of three fundamental factors:
  1. **Cognitive weights of Basic Control Structures (BCSs)**
  2. **Number of inputs (Ni)**
  3. **Number of outputs (No)**

- The **cognitive weight** of software is the degree of difficulty or effort required to understand a software component based on its control structures

Basic Control Structures

1. Sequence Structures

   - A series of actions completed in a specific order.

```
public static void main(String[] args) {
    System.out.println("Step 1: Start the program.");
    System.out.println("Step 2: Get user input.");
    System.out.println("Step 3: Process data.");
    System.out.println("Step 4: Display result.");
    System.out.println("Step 5: End the program.");
}
```

Basic Control Structures

2. Branch Structures

- Executes certain code only when a condition is met.

```
public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();
        if (age >= 18) {
            System.out.println("You are eligible to vote.");
        } else {
            System.out.println("You are not eligible to vote.");
        }
}
```

Basic Control Structures

3. Iterative Structures

- Executes a code block repeatedly until a condition is met.

```
public static void main(String[] args) {
    for (int i = 1; i <= 5; i++) {
        System.out.println("Iteration " + i);
    }
}
```

Basic Control Structures

Category	BCS	Structure	W_i
Sequence	Sequence (SEQ)		1
Branch	If-then-[else] (ITE)		2
	Case (CASE)		3

Category	BCS	Structure	W_i
Iteration	For - do (R_i)		3
	Do - while (R_1)		3
	While-do (R_0)		3

Category	BCS	Structure	W_i
Embedded component	Function call (FC)		2
	Recursion (REC)		3

Total Cognitive Weight

- The total cognitive weight of a software component, Wc is defined as the sum of the cognitive weights of its q linear blocks composed of the individual BCSs.

- Since each block may consist of m layers of nesting of BCSs, and each layer of n linear BCSs, Wc is calculated as follows

$$W_c = \sum_{j=1}^{q} \left[\prod_{k=1}^{m} \sum_{i=1}^{n} W_c(j,k,i) \right]$$

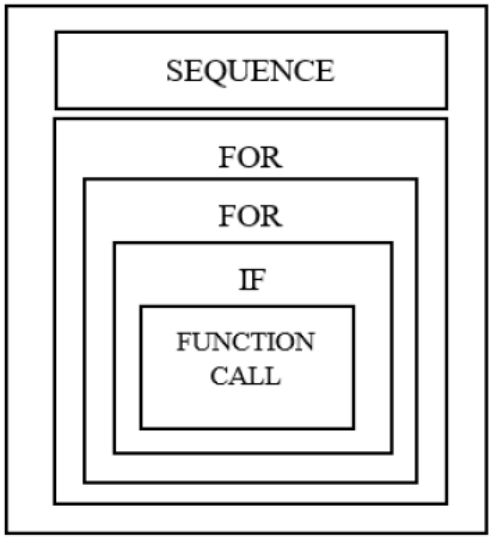
Total Cognitive Weight

- If there is no embedded BCS in any of the q blocks, i.e m = 1 then the previous equation can be simplified as follows

$$W_c = \sum_{j=1}^{q} \sum_{i=1}^{n} W_c(j,i)$$

Calculating Total Cognitive Weight

Source Code	Structure of BCSs	Cognitive Weight

public void bubbleSort(){
int out, in;
for(out=nElems-1; out>1; out--)
for(in=0; in<out; in++)
if(a[in] > a[in+1])
swap(in, in+1);
}
		$W_C = \sum_{j=1}^{q}[\prod_{k=1}^{m}\sum_{i=1}^{n} W_c\,(j,k,i)]$
		$W_C = 1 + 3\,(\,3\,(\,2\,(\,2\,)\,)\,)$
		$W_C = 1 + 36$
		$W_C = 37$

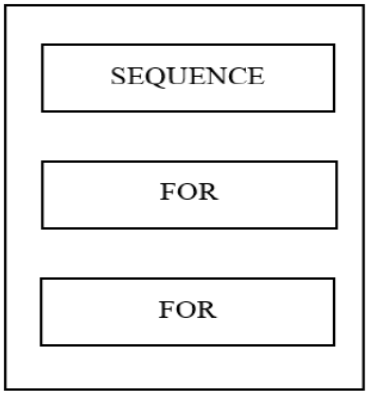
Calculating Total Cognitive Weight

Calculate the Total Cognitive Weight (Wc) for the below program

```
public static void main(String[] args) {

  String[] modules = {"SEPQM", "DS", "ESD", "AF", "SA"};

  for (int i = 0; i < modules.length; i++) {
        System.out.println(modules[i]);
  }

  System.out.println("In reverse order:");

  for (int i = modules.length - 1; i >= 0; i--) {
        System.out.println(modules[i]);
  }
}
```

Calculating Total Cognitive Weight

Source Code	Structure of BCSs	Cognitive Weight

public static void main(String[] args) {
String[] modules = {"SEPQM", "DS", "ESD", "AF", "SA"};
for (int i = 0; i < modules.length; i++) {
System.out.println(modules[i]);
}
System.out.println("In reverse order:");
for (int i = modules.length - 1; i >= 0; i--) {
System.out.println(modules[i]);
}
}
		$W_c = \sum_{j=1}^{q} \sum_{i=1}^{n} W_c(j,i)$
		$W_c = 1 + 3 + 3$
		$W_c = 7$

Cognitive Functional Size of a Basic Component

The cognitive functional size of a basic software component that only consists of one method Sf is defined as a product of the sum of inputs and outputs (Ni/No) and the total cognitive weight (Wc). More formally, it can be defined as follows

$$S_f = (N_i + N_o) \times W_c$$

Cognitive Functional Size of a Complex Component

The cognitive functional size of a complex software component with n methods Sf(c) is defined as follows

$$S_f(c) = \sum_{c=1}^{n} S_f(c)$$

Cognitive Functional Size of a Software System

The cognitive functional size of a component based software system, Ŝ, with p components Ŝf is defined as follows

$$\hat{S}_f = \sum_{k=1}^{p} S_f(k)$$

Calculate the Cognitive Functional Size

```
import java.util.Scanner;
public class Results{
  public static void main(String[] args) {
    System.out.print("Enter your marks: ");
    Scanner sc = new Scanner(System.in);
    int marks = sc.nextInt();
    while(marks < 0 || marks > 100){
      System.out.print("Enter a valid mark: ");
      marks = sc.nextInt();
    }
    if (marks>75)
      System.out.println("A Pass");
    else if (marks<=75 && marks>65)
      System.out.println("B Pass");
    else if (marks<=65 && marks>45)
      System.out.println("C Pass");
    else
      System.out.println("Fail");
  }
}
```

Calculate the Cognitive Functional Size

Source Code	Structure of BCSs	Cognitive Functional Size

import java.util.Scanner;
public class Results{
public static void main(String[] args) {
System.out.print("Enter your marks: ");
Scanner sc = new Scanner(System.in);
int marks = sc.nextInt();
while(marks < 0 || marks > 100){
System.out.print("Enter a valid mark: ");
marks = sc.nextInt();
}
if (marks>75)
System.out.println("A Pass");
else if (marks<=75 && marks>65)
System.out.println("B Pass");
else if (marks<=65 && marks>45)
System.out.println("C Pass");
else
System.out.println("Fail");
}
}
	SEQUENCE
	WHILE
	IF
	ELSE IF
	ELSE IF
		$W_c = \sum_{j=1}^{q}[\prod_{k=1}^{m}\sum_{i=1}^{n} W_c(j,k,i)]$
		$W_c = 1 + 3 + 2 + 2 + 2$
		$W_c = 10$
		
		$N_i = 2$
		$N_o = 1$ (Only one S.O.P statement is excuted at a given time)
		
		$S_f = (N_i + N_o) \times W_c$
		$S_f = (2 + 1) \times 10$
		$S_f = 30$ [CWU]

THANK YOU!