

Design Characteristics and Metrics: Part II

Hui Chen ^a

^aCUNY Brooklyn College, Brooklyn, NY, USA

May 9, 2023

Outline

- 1 Design Complexity
- 2 Cohesion and Coupling
- 3 Object-Oriented Complexity Metrics
- 4 References

Outline

- 1 Design Complexity
- 2 Cohesion and Coupling
- 3 Object-Oriented Complexity Metrics
- 4 References

Characterizing Design Complexity

- ▶ Halstead metrics
- ▶ McCabe's Cyclomatic Complexity metric (most broadly used)
- ▶ Henry-Kafura Information Flow (Fan-in/Fan-out) metrics
- ▶ Card and Glass design complexity metrics

Halstead Metrics

Measures the lexical complexity, rather than structural complexity of source code

- ▶ Use four fundamental units of measurements from code:
 - ▶ n_1 = number of distinct operators
 - ▶ n_2 = number of distinct operands
 - ▶ N_1 = sum of all occurrences of operators
 - ▶ N_2 = sum of all occurrences of operands
- ▶ Define:
 - ▶ Program vocabulary: $n = n_1 + n_2$
 - ▶ Program length: $N = N_1 + N_2$
- ▶ Compute 4 metric:
 - ▶ Volume: $V = N \log_2 n$
 - ▶ Potential volume: $V' = (2 + n'_2) \log_2(2 + n'_2)$ where n'_2 based on most "succinct" program's n_2 .
 - ▶ Program Implementation Level: $L = V'/V$
 - ▶ Effort: $E = V/L$

McCabe's Cyclomatic Complexity

Complexity of the program “control flow,” defined as

$$\text{Cyclomatic complexity} = E - N + 2p$$

where

- ▶ E = number of edges
- ▶ N = number of nodes
- ▶ p = number of connected components (usually 1)

McCabe's Cyclomatic Complexity: Example

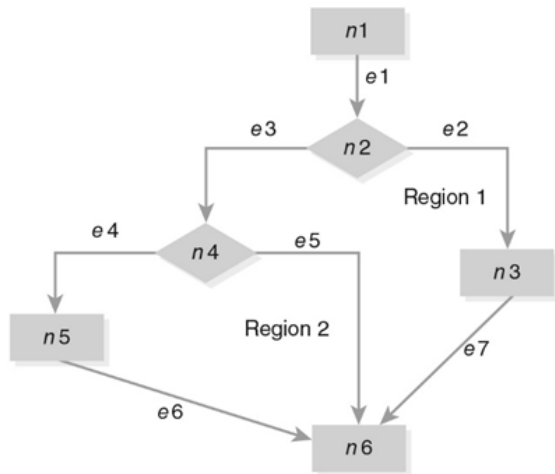


FIGURE 8.1 A simple flow diagram for cyclomatic complexity.

Henry-Kafura (Fan-in and Fan-out)

metric measures the inter-modular flow, defined as,

$$Cp = (\text{fan-in} \times \text{fan-out})^2$$

where “flows” concern

- ▶ Parameter passing
- ▶ Global variable access
- ▶ Inputs
- ▶ Outputs

Henry-Kafura (Fan-in and Fan-out) Complexity: Example

Assume:

- ▶ Fan-in, number of inter-modular flow into a program: 3
- ▶ Fan-out: number of inter-modular flow out of a program: 1

$$C_p = (3 \times 1)^2 = 9$$

Card and Glass Complexity

Also metric measures the inter-modular flow.

- ▶ Structural complexity of module x : $S_x = (\text{fan-out}_x)^2$
- ▶ Data complexity: $D_x = P_x / (\text{fan-out}_x + 1)$, where P_x is the number of variables passed to and from the module
- ▶ System complexity: $C_x = S_x + D_x$

Note “fan-in” is only a factor for the data complexity here.

Outline

- 1 Design Complexity
- 2 Cohesion and Coupling**
- 3 Object-Oriented Complexity Metrics
- 4 References

Cohesion and Coupling

- ▶ Cohesion: “degree of relatedness” within a unit, a module, an object, or a component. Higher \equiv Better
- ▶ Coupling: “degree of interdependence” between software units, modules, or components. Lower \equiv Better

Bieman and Ott's Functional Cohesion Metrics

Begin with counting:

- ▶ Data token: any occurrence of variable or constant in the program.
- ▶ Program slice: within a program, the collection of all the statements that can affect the value of some specific variable of interest.
- ▶ Data slice: within a program, the collection of all the data tokens in the slice that will affect the value of a specific variable of interest.
- ▶ Glue tokens: the data tokens in the program that lie in more than one data slice.
- ▶ Super glue tokens: the data tokens in the program that lie in every data slice of the program

Compute metrics:

- ▶ Weak functional cohesion:
 $(\# \text{ of glue tokens}) / (\text{total } \# \text{ of data tokens})$
- ▶ Strong functional cohesion:
 $(\# \text{ of super glue tokens}) / (\text{total } \# \text{ of data tokens})$

Bieman and Ott's Functional Cohesion Metrics: Example

Finding the maximum and minimum values procedure

```
MinMax (z, n)
integer end, min, max, i;
end = n,
max = z[0];
min = z[0];
For (i=1, i < end; i++){
  if z[i] > max then max = z[i];
  if z[i] > min then min = z[i];
}
return max, min;
```

Data Tokens:	Slice max:	Slice min:	Glue Tokens:	Superglue:
z1	z1	z1	z1	z1
n1	n1	n1	n1	n1
end1	end1	end1	end1	end1
min1	max1	min1	11	11
max1	11	11	end2	end2
11	end2	end2	n2	n2
end2	n2	n2	12	12
n2	max2	min2	03	03
max2	z2	z3	13	13
z2	01	02	end3	end3
01	12	12	14 (11)	14 (11)
min2	03	03		
z3	13	13		
02	end3	end3		
12	14	14		
03	z4	z6		
13	15	17		
end3	max3	min3		
14	max4	min4		
z4	z5	z7		
15	16	18		
max3	max5	min5		
max4	(22)	(22)		
z5				
16				
z6				
17				
min3				
min4				
z7				
18				
max5				
min5 (33)				

FIGURE 8.2 A pseudocode example of functional cohesion measures.

Bieman and Ott's Functional Cohesion Metrics: Example

Let end be 5. The glue tokens are the same as the super glue tokens.

- ▶ Super glue tokens = 11
- ▶ Glue tokens = 11

The data slice for min and data slice for max are also the same here: 22.
The total number of data tokens: 33.

The cohesion metrics for the example of min-max are:

- ▶ Weak functional cohesion = $11 / 33 = 1/3$
- ▶ Strong functional cohesion = $11 / 33 = 1/3$

What if we refactored the minmax function to two functions, Max and Min?

What if we increase the value of end

Outline

- 1 Design Complexity
- 2 Cohesion and Coupling
- 3 Object-Oriented Complexity Metrics**
- 4 References

Chidamber and Kemerer (C-K) OO Metrics

- ▶ Weighted Methods per Class (WMC)
- ▶ Depth of Inheritance Tree (DIT)
- ▶ Number of Children (NOC)
- ▶ Coupling Between Object Classes (CBO)
- ▶ Response for a Class (RFC)
- ▶ Lack of Cohesion in Methods (LCOM)

Lack of Cohesion of Methods (LCOM)

High LCOM indicates low cohesion and possibly high complexity. A simple method to estimate LCOM,

$$LCOM = 1 - \frac{\sum(m_i)}{MV} \quad (1)$$

where

- ▶ $LCOM \in [0, 1]$
- ▶ $M = \#$ of methods of the class
- ▶ $V = \#$ of instance or class variables of the class
- ▶ $m_i = \#$ of class methods that access the i 'th class variable

Observations:

- ▶ A class is utterly cohesive if all its methods use all its instance fields, because $\sum(m_i) = MV$ and then $LCOM = 0$
- ▶ High LCOM suggests possible violation of the Single Responsibility Principle

Summary and Questions

An introduction to several complexity metrics.

- ▶ Halstead metrics
- ▶ McCabe's Cyclomatic Complexity metric (most broadly used)
- ▶ Henry-Kafura Information Flow (Fan-in/Fan-out) metrics
- ▶ Card and Glass design complexity metrics
- ▶ Cohesion metrics
- ▶ OO metrics, in particular, LCOM

Let's do an exercise ...

Outline

- 1 Design Complexity
- 2 Cohesion and Coupling
- 3 Object-Oriented Complexity Metrics
- 4 **References**

“Engineering Software as a Service” by Armando Fox and David Patterson
(2nd Edition)

“[Introduction to Software Design with Java](#)” by Martin P. Robillard

“Essentials of Software Engineering” by Frank Tsui, Orlando Karam, and
Barbara Bernal(4th Edition)