

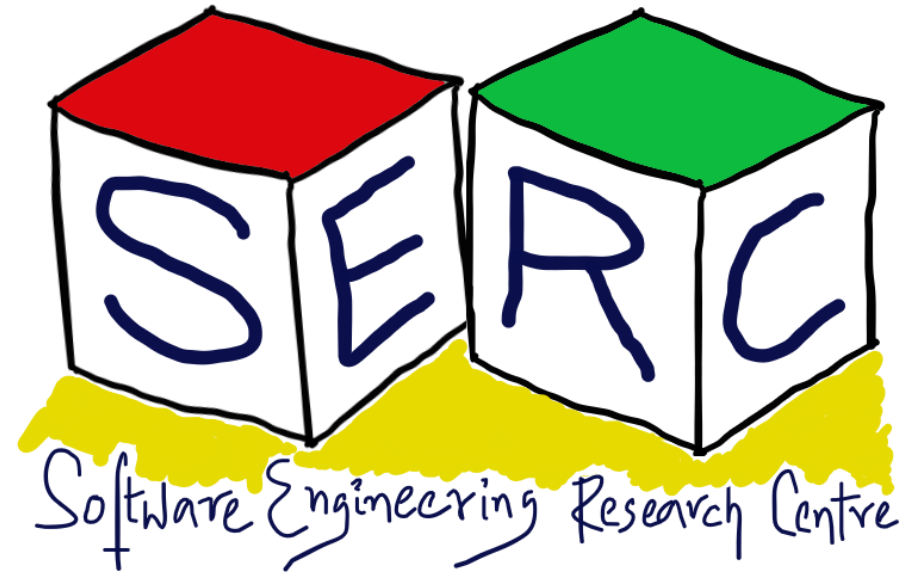
Code Metrics

CS6.401 Software Engineering

Dr. Karthik Vaidhyanathan

karthik.vaidhyanathan@iiit.ac.in

<https://karthikvaidhyanathan.com>



INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY

HYDERABAD



Can some metrics be used to
aid refactoring?

Code Complexity

The ratio of time spent reading versus writing is well over 10 to 1

--Robert C Martin

- Code over time has tendency to accumulate complexity
- Greater or larger functionality should not have direct impact on code complexity
- Unnecessary complexity affects maintainability, time to market, understandability and testability

How to manage it? – Start measuring it!!

What is measurement?

*Measurement is defined as the **process** by which **numbers or symbols are assigned to attributes of entities** in the real world in such a way as to **describe them** according to clearly **defined rules***

What is measurement?

- Entity: can be an Object (person) or event (journey)
- Attribute: Feature or property of entity (height, blood pressure, etc.)
- Two types of measurement:
 - Direct measurement: measurement of attribute
 - Indirect measurement: Measurement of attribute involves measurement of some other attribute (eg: BMI)
- Uses of measurement – Assessment or Prediction

Measurement In terms of Software

- Carried out throughout the software development process
- Measurements can be performed at different levels
 - Completed Product (reliability, performance, etc.)
 - Development Process (time, man hours, etc.)
 - Source Code (lines of code, cyclomatic complexity, etc.)
- Source code metrics focus on measuring the source code of a system
 - Allows to measure complexity of code
 - Improve quality of code and thereby overall software
 - Used for lot of applications (defect prediction, fault localizations, refactoring, testing, etc.)

Commonly Used Source Code Metrics

- Lines of Code (LOC)
 - Easiest but **effective indicator of complexity**
 - Small modules have low defect rates as opposed to large ones
- Cyclomatic Complexity
 - Developed by Thomas McCabe, 1976
 - Allows to measure the **complexity with respect to control flow of the code**
- Halstead Software Science Metrics
 - Developed by Halstead, 1977
 - Measures **complexity in terms of the amount of information in source code**
- There are also object oriented metrics (Chidamber and Kemerer 1994, Li and Henry 1993)

Cyclomatic Complexity

- Count of the number of linearly independent paths in a program
- Has a big impact on testing – test cases needs to cover the different paths
- Uses the control flow graph, G of the given program – Approach based on graph theory
- $V(G) = e - n + 2p$
 - e = Number of edges
 - n = Number of nodes
 - p = Connected components

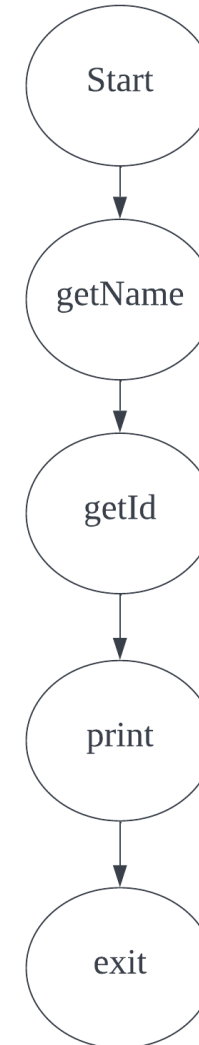
In practice the number boils down to 1 (base) + number of decision points

Cyclomatic Complexity - Simple Example

```
Display Student

1 public void displayDetails(Student student)
2 {
3     name = student.getName();
4     id = student.getId();
5     System.out.println(name + " " + id);
6 }
```

$$\begin{aligned}\text{Complexity} &= 4 - 5 + 2*1 \\ &= 1\end{aligned}$$

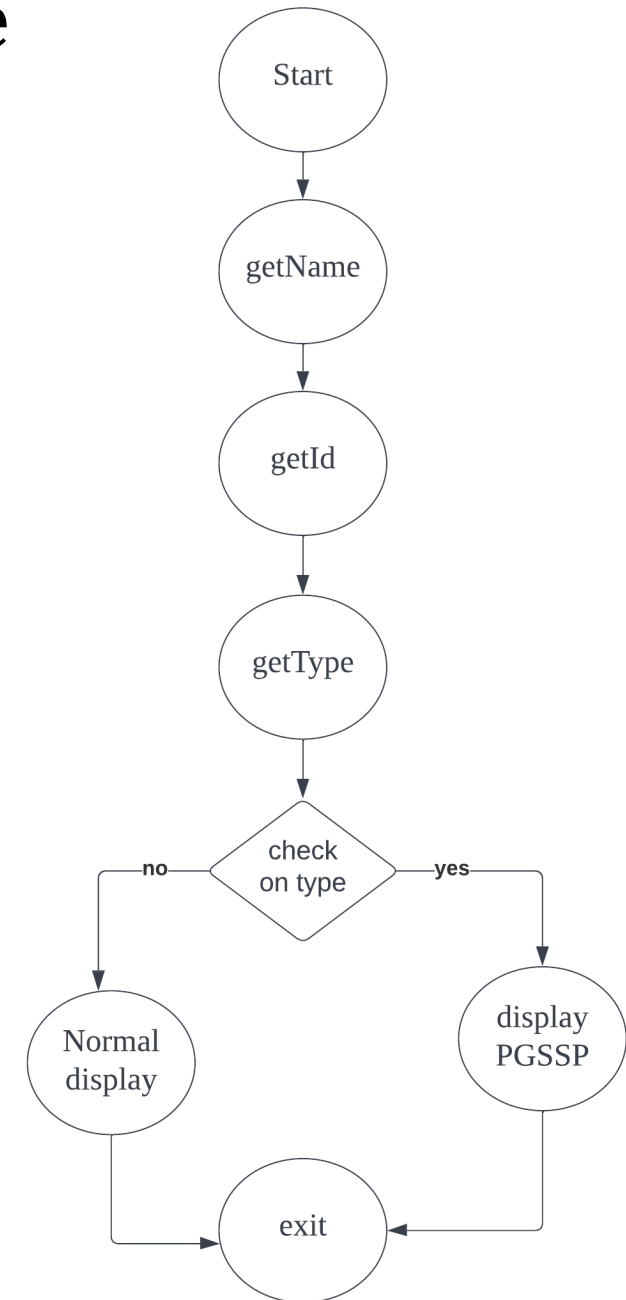


Cyclomatic Complexity - Another Example

```
Highlight PGSSP Students

1 public void displayDetails(Student student)
2 {
3     name = student.getName();
4     id = student.getId();
5     type = student.getType();
6     if (type.equals("PGSSP"))
7     {
8         System.out.println(name + " " + id + " " + "PGSSP");
9     }
10    else
11    {
12        System.out.println(name + " " + id);
13    }
14 }
```

$$\begin{aligned}\text{Complexity} &= 8 - 8 + 2*1 \\ &= 2\end{aligned}$$



Halstead Software Science Metrics

- Considers program as a collection of tokens
- Tokens: Operators or operands
- The metrics makes use of the occurrence of operators and operands in a program to reason about complexity

n1 -> number of distinct operators (+, -, *, while, for, (), {}, function calls, etc.)

n2 -> number of distinct operands (variables, method names, etc.)

N1 -> total number of occurrence of operators

N2 -> total number of occurrence of operands

- The above observations are combined to provide different metrics

Halstead Software Science Metrics

- Vocabulary, $n = n_1 + n_2$
- Program length $N = N_1 + N_2$
- Volume, $V = N \log_2 (n)$
-

Operators (+, *, =, double, int, final, return, {, }, (,)), $n_1 = 11$

Operands (calculateTotalCost, item1, item2, sum, tax, number1, number 2, totalCost) = 8

$N_1 - (1, 1, 3, 3, 3, 1, 1, 1, 1, 1, 1) = 17$ $n = 19, N = 28, V = 28 \log_2(19) = 35.80$

$N_2 - (1, 1, 1, 2, 2, 1, 1, 2) = 11$

```
Simple Sum function

1 public double calculateTotalCost(int item1, int item2)
2 {
3     int sum;
4     final double tax = 0.12;
5     sum = number1 + number2;
6     double totalCost = sum*tax;
7     return totalCost;
8 }
```

Six OO Metrics – Chidamber and Kemerer

- Weighted Methods per Class
- Depth of Inheritance Tree
- Number of Children of a Class
- Coupling Between Object Classes
- Response for a Class
- Lack of Cohesion on Methods

Thank You



Course website: karthikv1392.github.io/cs6401_se

Email: karthik.vaidhyanathan@iiit.ac.in

Web: <https://karthikvaidhyanathan.com>

Twitter: @karthi_ishere

