Software Engineering(303105253)

Dr. Kapil Aggarwal, Associate Professor
Computer Science & Engineering

Module-5

Coding and Unit Testing: Programming principles and guidelines, Programming practices, Coding standards, Incremental development of code, Management of code evaluation, Unit testing- procedural units, classes, Code Inspection, Metrics- size measure, complexity metrics, Cyclomatic Complexity, Halstead measure,Knot Count, Comparison Of Different Metrics

TESTING

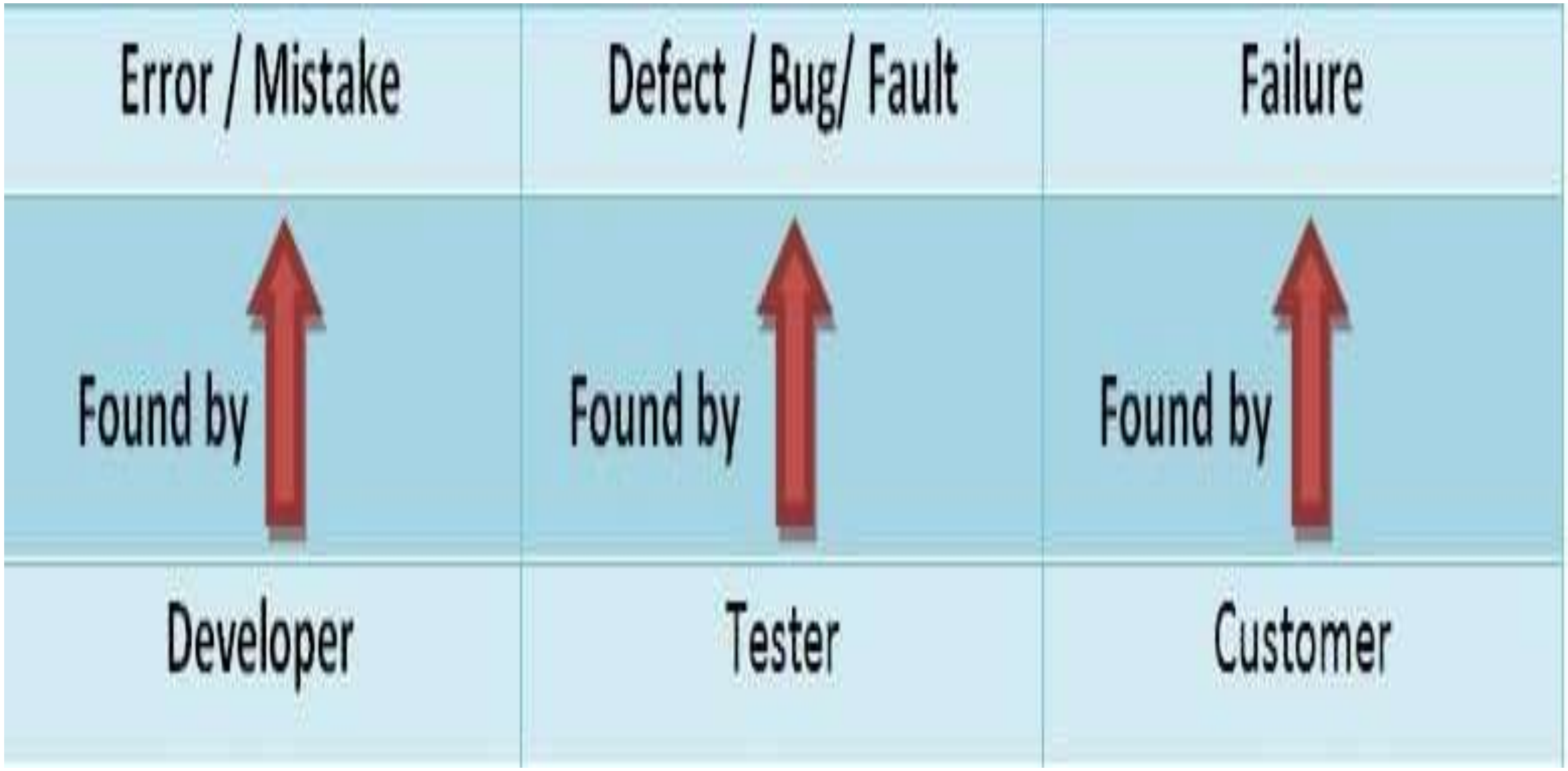
Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the system is defect free.

- A Person Makes an Error.
- That creates a fault in Software.
- That can cause a Failure in operation

Error: An error is a human action that produces the incorrect result that results in a fault.

Bug: the presence of error at the time of execution of software

Failure: Deviation of the software from its expected result, It is an event.



Error / Mistake	Defect / Bug/ Fault	Failure
Found by ↑	Found by ↑	Found by ↑
Developer	Tester	Customer

CODING STANDARDS

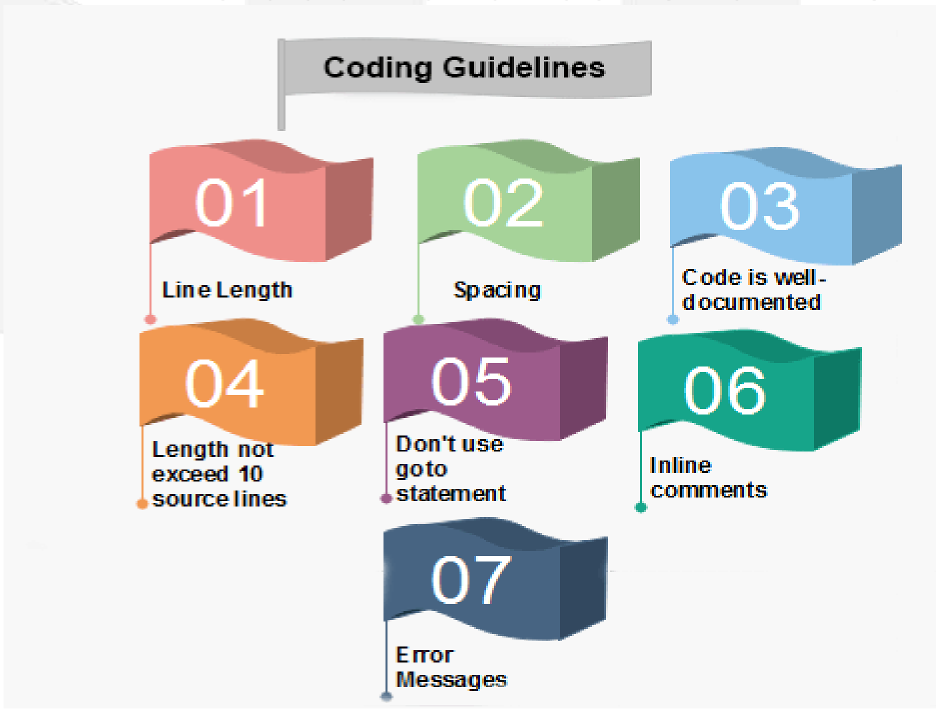
- System design transformed into implementation model.
- Objectives of Coding Standards are:
- Programs developed in coding should be in readable.
- Should execute efficiently.
- The program utilize less amount of time.
- To maintain some well defined and standard style of coding.

What are coding rules and guidelines?

- Safe: it can be used without harm
- Secure
- Reliable
- Testable
- maintainable

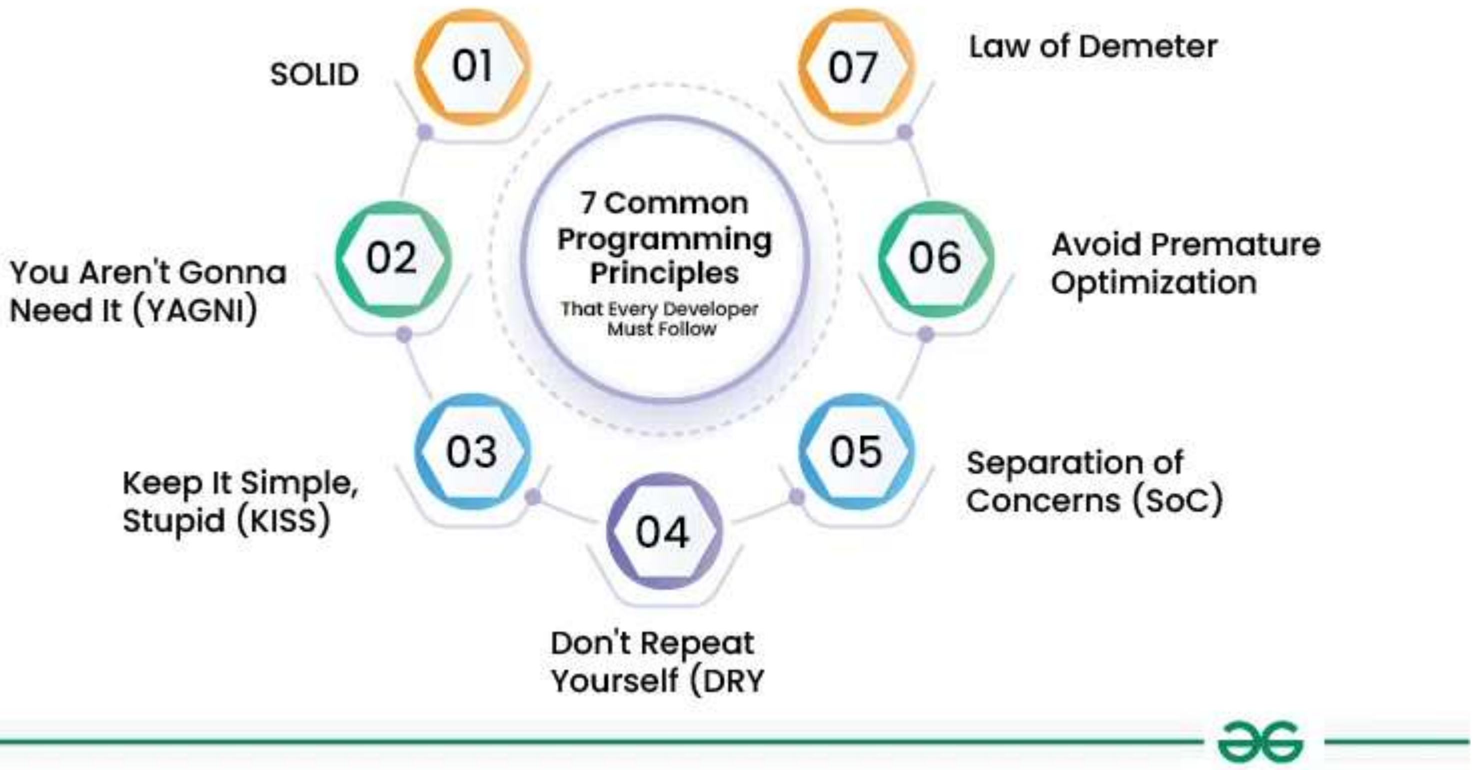
Programming Principles and Guidelines

The main task before a programmer is to **write quality code with few bugs in it**. The additional constraint is to write code quickly.



1. Line Length: It is considered a good practice to keep the length of source code **lines at or below 80 characters**. Lines longer than this may not be visible properly on some terminals and tools. Some printers will truncate lines longer than 80 columns.
2. Spacing: The appropriate use of spaces within a line of code can improve readability.
3. The code should be well-documented: As a rule of thumb, there must be at least one comment line on the average for every three-source line.
4. The length of any function should not exceed 10 source lines: A very lengthy function is generally very difficult to understand as it possibly carries out many various functions. For the same reason, lengthy functions are possible to have a disproportionately larger number of bugs.
5. Do not use goto statements: Use of goto statements makes a program unstructured and very tough to understand.

Programming Principles


7 Common Programming Principles
That Every Developer Must Follow
01
SOLID
02
You Aren't Gonna Need It (YAGNI)
03
Keep It Simple, Stupid (KISS)
04
Don't Repeat Yourself (DRY
05
Separation of Concerns (SoC)
06
Avoid Premature Optimization
07
Law of Demeter

4. This principle is an acronym of the five principles which is given below**. SOLID**

- **S**ingle Responsibility Principle (SRP)

  In an application that calculates the **area of shapes**, you can create a class for each shape and a separate class to calculate the area of all the shapes.
- **O**pen/Closed Principle**:**

  a software design principle that encourages adding new features to a system without modifying existing code

  Eg. credit card payments - Paypal
- **L**iskov's Substitution Principle (LSP)

  objects of a superclass should be replaceable with objects of its subclasses without affecting the correctness of the program.

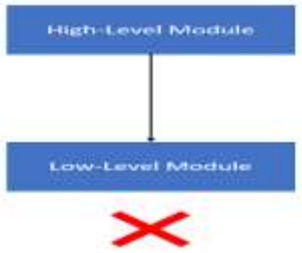
- **Interface Segregation Principle (ISP)**

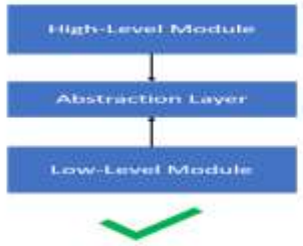
  do not force any client to implement an interface which is irrelevant to them.Eg. Menu Card

- **Dependency Inversion Principle (DIP)**

  High-level modules should not depend on low-level modules. Both should depend on abstractions.

  Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.

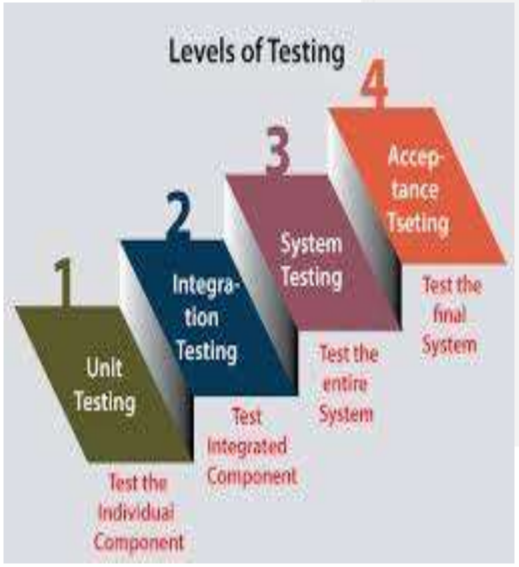


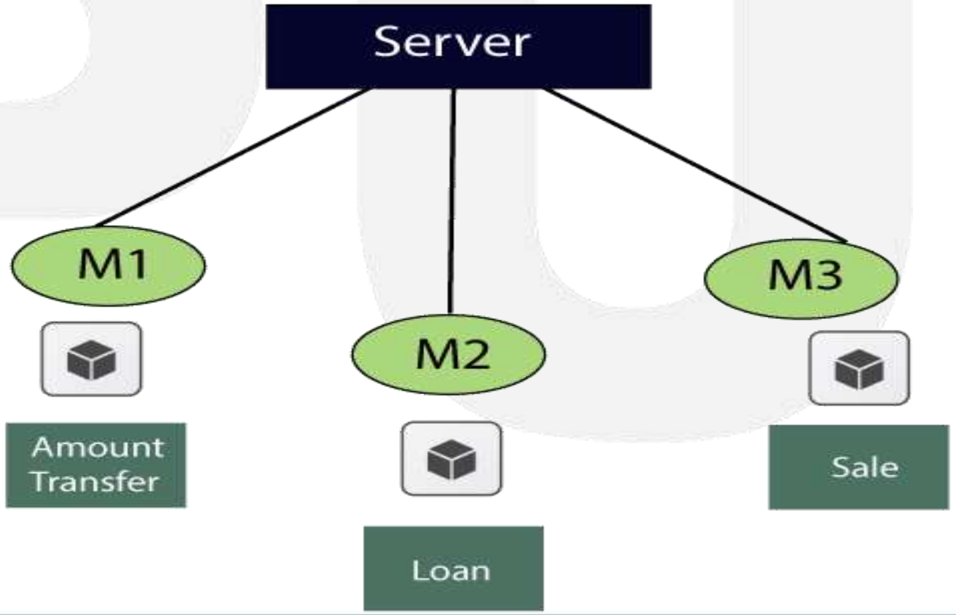


1. Nobody in programming loves to debug, maintain, or make changes in complex code. It is very first principle, acronym stands for **Keep It Simple, Stupid**.
2. ***Don't Repeat Yourself (DRY)"*** principal goal is to reduce the **repetition of code**. lines of code will be run multiple times.
3. ***"You Aren't Gonna Need It (YAGNI)"*** principle states that "don't implement something until it is necessary" because in most of the cases **you are not going to use that piece of code** in future.

Unit testing

Unit Testing is a software testing technique in which individual units or components of a software application are tested in isolation.



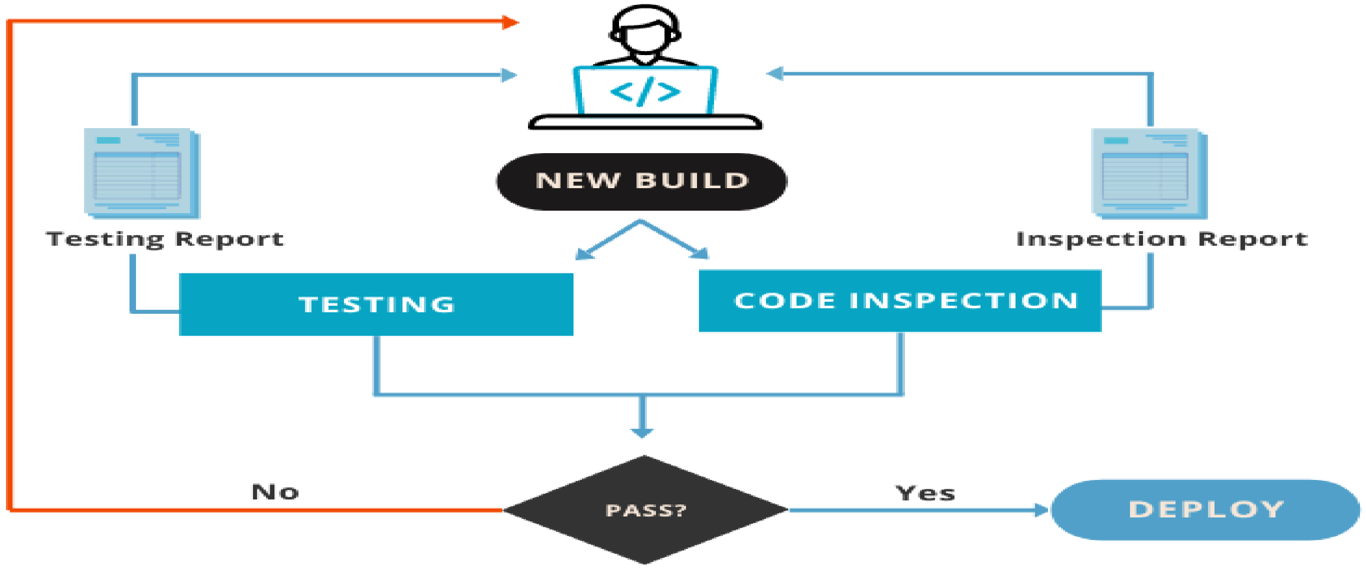


Parul® University

DIGITAL LEARNING CONTENT

Code Inspection

- Code Inspection is the process of **reviewing and improving the code before testing.**
- Code inspection analyzes the source code based on a set of rules.



Advantages

- Engineers can enhance the quality of the product.
- Reviewing the code easily identifies issues.
- Improving team collaboration
- Improving code efficiency

Software Inspection Process

Planning : The moderator plan the inspection.
Overview Meeting: The background of the work product is described by the author.
Preparation: The examination of the work product is done by inspector to identify the possible defects.
Inspection Meeting: The reader reads the work product part by part during this meeting and the inspectors the faults of each part.
Rework: After the inspection meeting, the writer changes the work product according to the work plans.
Follow Up: The changes done by the author are checked to make sure that everything is correct.

Metrics

Metrics measure the size and complexity of code.

Cyclomatic complexity

A high cyclomatic complexity means the code is difficult to understand and maintain.

Halstead metrics

Measures the volume and difficulty of understanding a program's code. Halstead metrics can **be used to compare the complexity of two programs.**

Size-oriented metrics

Measure the **size of a program and the development process**. Examples include the number of lines of code, time spent, and money spent.

Metrics- Metrics measure the size and complexity of code.

Cyclomatic Complexity:

- Cyclomatic complexity is a software metric used to measure the complexity of a program. Thomas J. McCabe developed this metric in 1976.

How to Calculate Cyclomatic Complexity?

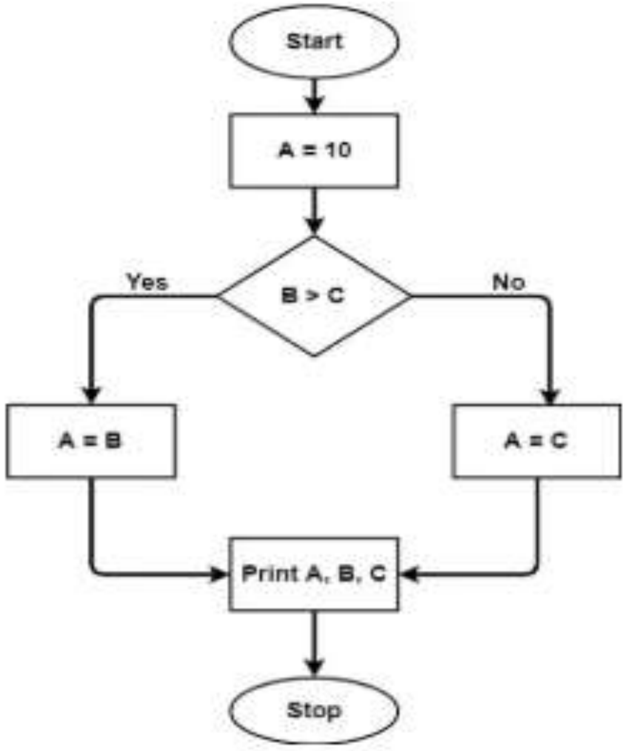
For a program control graph G, cyclomatic number, V (G), is given as:

$$V(G) = E - N + 2 * P$$

E = The number of edges in graphs G

N = The number of nodes in graphs G

P = The number of connected components in graph G.



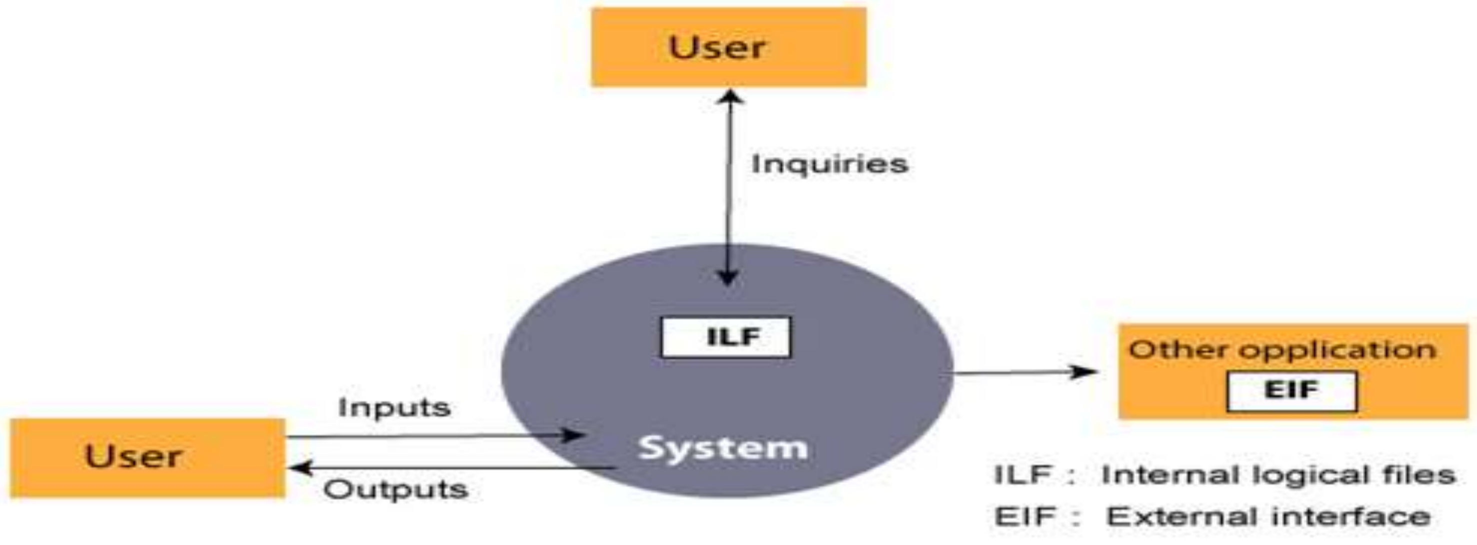
Cyclomatic Complexity

The graph shows seven shapes(nodes), and seven lines(edges), hence cyclomatic complexity is 7-7+2 = 2.

Function Point analysis(FPA)

- ✓ FPA is used to make estimate of the software project, including its testing in terms of functionality or function size of the software product.
- ✓ However, functional point analysis may be used for the test estimation of the product.

The FPA functional units are shown in Fig:



FPAs Functional Units System

Measurements Parameters	**Examples**
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

Example: Compute the function point, productivity, documentation, cost per function for the following data:

Number of user inputs = 24

Number of user outputs = 46

Number of inquiries = 8

Number of files = 4

Number of external interfaces = 2

Effort = 36.9 p-m

Technical documents = 265 pages

User documents = 122 pages

Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Solution:

Measurement Parameter	Count		Weighing factor
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

So sum of all fi (i ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = Count-total * [0.65 + 0.01 *∑(fi)]
= 378 * [0.65 + 0.01 * 43]
= 378 * [0.65 + 0.43]
= 378 * 1.08 = 408

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation = technical document + user document
= 265 + 122 = 387pages

Documentation = Pages of documentation/FP
= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$

Halstead measure:

A computer program is an implementation of an algorith considered to be **a collection of tokens which can be classifie as either operators or operand.“**

Token Count

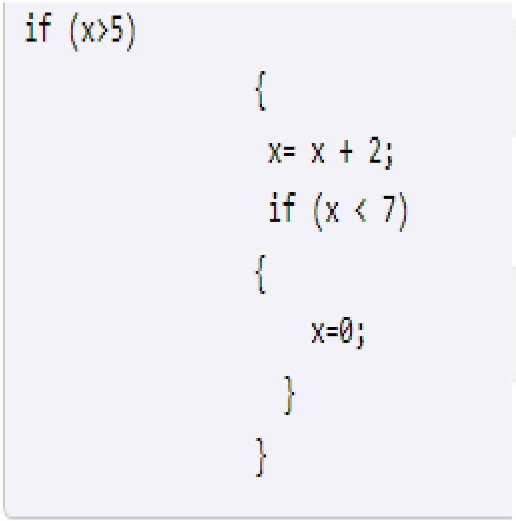
All software science metrics can be defined in terms of these basic symbols. These symbols are called as a token.

[redacted]

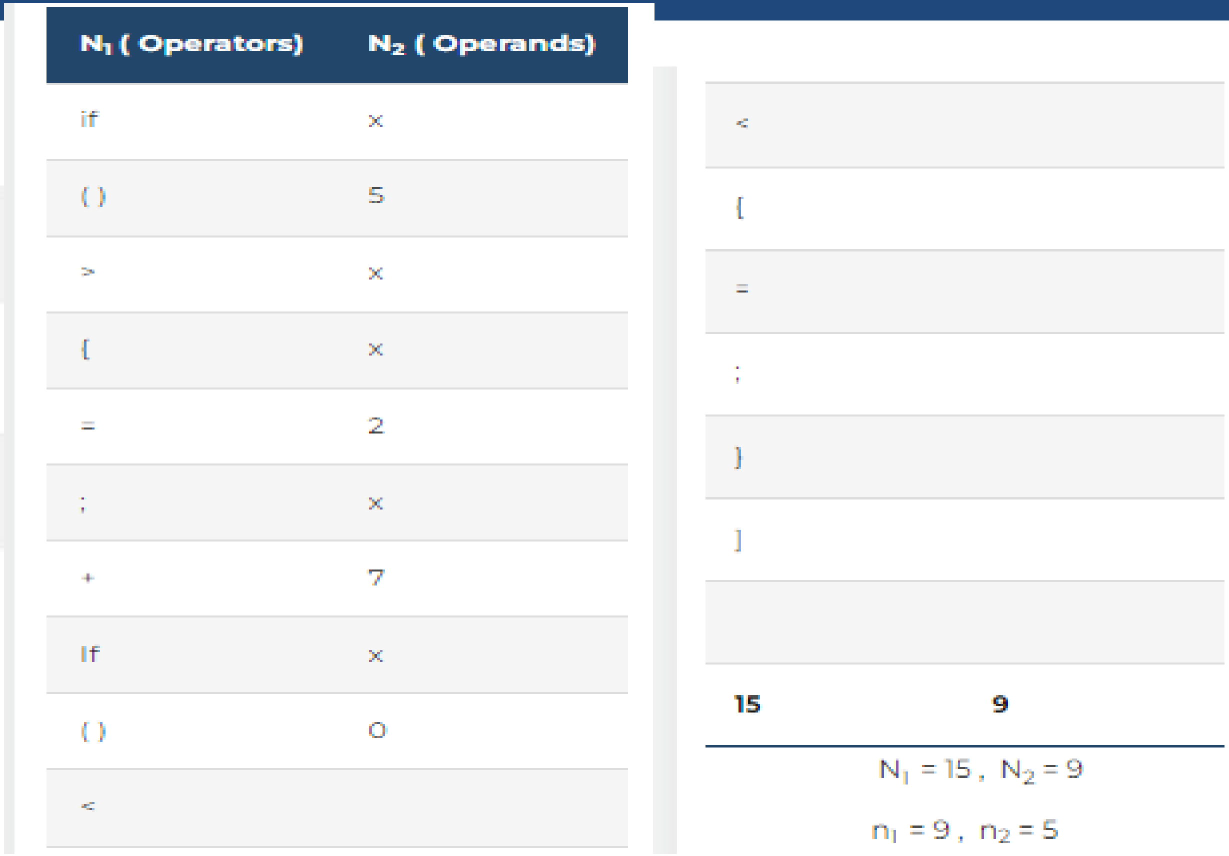
- Program vocabulary: $\eta = \eta_1 + \eta_2$
- Program length: $N = N_1 + N_2$
- Calculated estimated program length: $\hat{N} = \eta_1 \log_2 \eta$
- Volume: $V = N \times \log_2 \eta$
- Difficulty : $D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2}$
- Effort: $E = D \times V$

[redacted]

Example



```
if (x>5)
            {
             x= x + 2;
             if (x < 7)
            {
               x=0;
              }
            }
```



N_1 (Operators)	N_2 (Operands)
if	x
( )	5
>	x
{	x
=	2
;	x
+	7
If	x
( )	0
<	
<	
{	
=	
;	
}	
}	
15	**9**

$N_1 = 15$, $N_2 = 9$

$n_1 = 9$, $n_2 = 5$

Total Length (N)

$$N = N_1 + N_2 = 15 + 9 = 24$$

Vocabulary (n)

$$n = n_1 + n_2 = 9 + 5 = 14$$

N/ N

$$\mathbf{N = n_1 \log_2 n_1 + n_2 \quad \log_2 n_2}$$

$$= 9 \log_2 9 + 5 \log_2 5$$

$$= 9 \log_{10} 9 / 0.3010 + 5 \log_{10} 5 / 0.3010$$

$$= 9 \times (3.17) + 5 \times (2.32)$$

$$= 28.53 + 116$$

$$= 40.13$$

Estimated Length = N/N = 40.13/ 24 = 1.67

Volume = N $\log_2$ n

= 24 $\log_2$ 14

= 24 x $\log_{10}$ 14 / 0.3010

= 24 x 3.807 = 91.36

D = N_1/ 2 x N_2 / n_2

= 9/2 x 9/5 = 81 /10 = 8.1

Effort = V *D

= 91.36 x 8.1 = 740.016

The basic measures are
n1 = count of unique operators.
n2 = count of unique operands.
N1 = count of total occurrences of operators.
N2 = count of total occurrence of operands.
In terms of the total tokens used, the size of the program can be expressed as N = N1 + N2.

Knot Count

- A **knot** is essentially the intersection of two such control transfer arrows. If each statement in the program is written on a separate line, this notion can be formalized as follows.
  - A jump from the line **a** to line **b** is represented by the **pair(a, b)**.
  - Two jumps **(a, b)** and **(p, q)** give rise to a **knot** if either
  - **min (a, b) < min (p, q) < max (a, b)** and **max (p, q) > max (a, b)** or **min (a, b) < max (p, q) < max (a, b)** and **min (p, q) < min (a, b)**.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in