

# Software Metrics

# Software Metrics

- ▶ IEEE defines metric as ‘a quantitative measure of the degree to which a system, component, or process possesses a given attribute.’
- ▶ The goal of software metrics is to identify and control essential parameters that affect software development
- ▶ Software metrics help project managers to gain an insight into the efficiency of the software process, project, and product

# Measures and Metrics

- ▶ Measure can be defined as quantitative indication of amount, dimension, capacity, or size of product and process attributes.
- ▶ Measurement can be defined as the process of determining the measure.
- ▶ Metrics can be defined as quantitative measures that allow software engineers to identify the efficiency and improve the quality of software process, project, and product.

# Classification of Software Metrics

## ► Process Metrics

- Process metrics assess the effectiveness and quality of software process, determine maturity of the process, effort required in the process, effectiveness of defect removal during development, and so on.

## ► Product Metrics

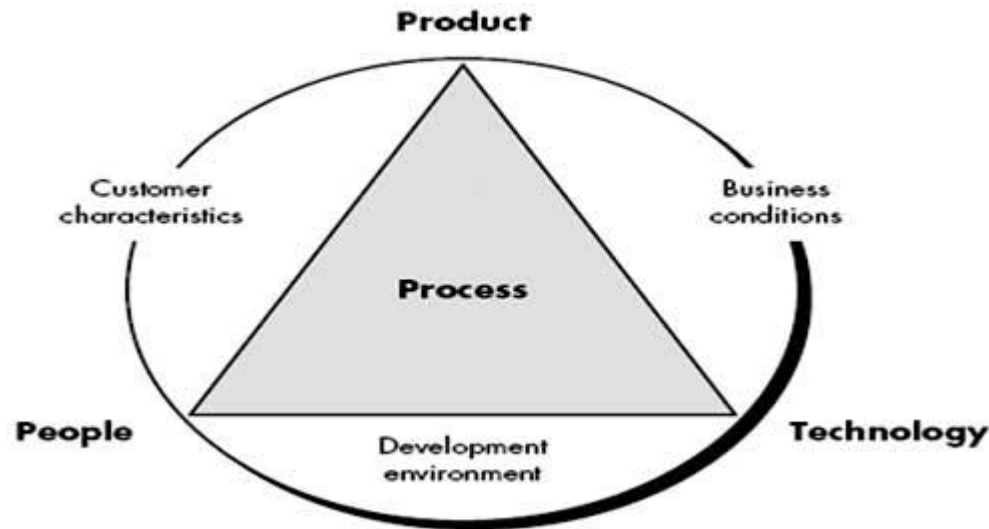
- Product metrics is the measurement of work product produced during different phases of software development
  - Metrics for the Analysis Model
  - Metrics for Software Design
  - Metrics for Coding
  - Metrics for Software Testing
  - Metrics for Software Maintenance


## ► Project Metrics

- Project metrics illustrate the project characteristics and their execution.

# Process metrics

- ▶ Using software process metrics, software engineers are able to assess the efficiency of the software process that is performed using the process as a framework.
- ▶ Process is placed at the centre of the triangle connecting three factors namely product, people, and technology which have an important influence on software quality and organization performance.



- 
- ▶ To measure the efficiency and effectiveness of the software process, a set of metrics is formulated based on the outcomes derived from the process. These outcomes are listed below.
    - ▶ Number of errors found before the software release
    - ▶ Defect detected and reported by the user after delivery of the software
    - ▶ Time spent in fixing errors
    - ▶ Work products delivered
    - ▶ Human effort used
    - ▶ Time expended
    - ▶ Conformity to schedule
    - ▶ Wait time
    - ▶ Number of contract modifications
    - ▶ Estimated cost compared to actual cost.

# Product metrics

- ▶ In software development process, a working product is developed at the end of each successful phase.
- ▶ Each product can be measured at any stage of its development.
- ▶ Metrics are developed for these products so that they can indicate whether a product is developed according to the user requirements.

# Metrics for the Analysis Model

- ▶ **Function Point (FP) Metric**
- ▶ **Lines of Code (LOC)**
- ▶ **Metrics for Specification Quality**



# Metrics for OOD

- ▶ **Complexity:** Determined by assessing how classes are related to each other
- ▶ **Coupling:** Defined as the physical connection between OO design elements
- ▶ **Sufficiency:** Defined as the degree to which an abstraction possesses the features required of it
- ▶ **Cohesion:** Determined by analyzing the degree to which a set of properties that the class possesses is part of the problem domain or design domain
- ▶ **Primitiveness:** Indicates the degree to which the operation is atomic
- ▶ **Similarity:** Indicates similarity between two or more classes in terms of their structure, function, behavior, or purpose
- ▶ **Volatility:** Defined as the probability of occurrence of change in the OO design
- ▶ **Size:** Defined with the help of four different views, namely, population, volume, length, and functionality. Population is measured by calculating the total number of OO entities, which can be in the form of classes or operations. Volume measures are collected dynamically at any given point of time. Length is a measure of interconnected designs such as depth of inheritance tree. Functionality indicates the value rendered to the user by the OO application.

# Metrics for Coding

- ▶ Halstead proposed the first analytic laws for computer science by using a set of primitive measures
  - ▶  $n_1$  = number of distinct operators in a program
  - ▶  $n_2$  = number of distinct operands in a program
  - ▶  $N_1$  = total number of operators
  - ▶  $N_2$  = total number of operands.
- ▶ By using these measures, Halstead developed an expression for overall program length, program volume, program difficulty, development effort, and so on.
- ▶ Program length (N) can be calculated by using the following equation.
  - ▶  $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$ .
- ▶ Program volume (V) can be calculated by using the following equation.
  - ▶  $V = N \log_2 (n_1 + n_2)$ .
- ▶ Program difficulty level (D) and effort (E) can be calculated by using the following equations.
  - ▶  $D = (n_1/2) * (N_2/n_2)$ .
  - ▶  $E = D * V$ .

# Metrics for Software Testing

- ▶ A set of metrics proposed for OO testing is listed below.
- ▶ **Lack of cohesion in methods (LCOM):** This indicates the number of states to be tested. LCOM indicates the number of methods that access one or more same attributes. The value of LCOM is 0, if no methods access the same attributes. As the value of LCOM increases, more states need to be tested.
- ▶ **Percent public and protected (PAP):** This shows the number of class attributes, which are public or protected. Probability of adverse effects among classes increases with increase in value of PAP as public and protected attributes lead to potentially higher coupling.
- ▶ **Public access to data members (PAD):** This shows the number of classes that can access attributes of another class. Adverse effects among classes increase as the value of PAD increases.
- ▶ **Number of root classes (NOR):** This specifies the number of different class hierarchies, which are described in the design model. Testing effort increases with increase in NOR.
- ▶ **Fan-in (FIN):** This indicates multiple inheritances. If value of FIN is greater than 1, it indicates that the class inherits its attributes and operations from many root classes. Note that this situation (where  $FIN > 1$ ) should be avoided.

# Metrics for Software Maintenance

- ▶ For the maintenance activities, metrics have been designed explicitly. IEEE have proposed Software Maturity Index (SMI), which provides indications relating to the stability of software product.
- ▶ For calculating SMI, following parameters are considered.
  - ▶ Number of modules in current release ( $M_T$ )
  - ▶ Number of modules that have been changed in the current release ( $F_e$ )
  - ▶ Number of modules that have been added in the current release ( $F_a$ )
  - ▶ Number of modules that have been deleted from the current release ( $F_d$ )
- ▶ Once all the parameters are known, SMI can be calculated by using the following equation.
  - ▶  $SMI = [M_T - (F_a + F_e + F_d)] / M_T$ .

# Project metrics

- ▶ Project metrics enable the project managers to assess current projects, track potential risks, identify problem areas, adjust workflow, and evaluate the project team's ability to control the quality of work products.
- ▶ Note that project metrics are used for tactical purposes rather than strategic purposes used by the process metrics.

# Test Metrics Life Cycle

| Different stages of Metrics life cycle | Steps during each stage  |
|--|--|
| •Analysis                              | <ul style="list-style-type: none"><li>•Identification of the Metrics</li><li>•Define the identified Metrics</li></ul>  |
| •Communicate                           | <ul style="list-style-type: none"><li>•Explain the need for metric to stakeholder and testing team</li><li>•Educate the testing team about the data points to need to be captured for processing the metric</li></ul>    |
| •Evaluation                            | <ul style="list-style-type: none"><li>•Capture and verify the data</li><li>•Calculating the metrics value using the data captured</li></ul>  |
| •Report                                | <ul style="list-style-type: none"><li>•Develop the report with an effective conclusion</li><li>•Distribute the report to the stakeholder and respective representative</li><li>•Take feedback from stakeholder</li></ul> |

# How to calculate Test Metrics

| Sr# | Steps to test metrics   | Example  |
|-----|---|--|
| 1   | Identify the key software testing processes to be measured  | •Testing progress tracking process   |
| 2   | In this Step, the tester uses the data as a baseline to define the metrics                          | •The number of test cases planned to be executed per day   |
| 3   | Determination of the information to be followed, a frequency of tracking and the person responsible | •The actual test execution per day will be captured by the test manager at the end of the day                                    |
| 4   | Effective calculation, management, and interpretation of the defined metrics                        | •The actual test cases executed per day  |
| 5   | Identify the areas of improvement depending on the interpretation of defined metrics                | •The <u>Test Case</u> execution falls below the goal set, we need to investigate the reason and suggest the improvement measures |