CSC 2210
Object Oriented Analysis & Design

Dr. Akinul Islam Jony

Associate Professor

Department of Computer Science, FSIT

American International University - Bangladesh (AIUB)

akinul@aiub.edu

Software Metrics Overview

>> What is Measurement?
>> Software Metrics Challenges
>> What is Software Measurement?
>> Scope of Software Metrics

What is Measurement?

>> **Measurement** is the process by which **numbers** or **symbols** are assigned to **attributes** of **entities** (**objects**) in the real world in such a way as to ascribe them according to defined rules.

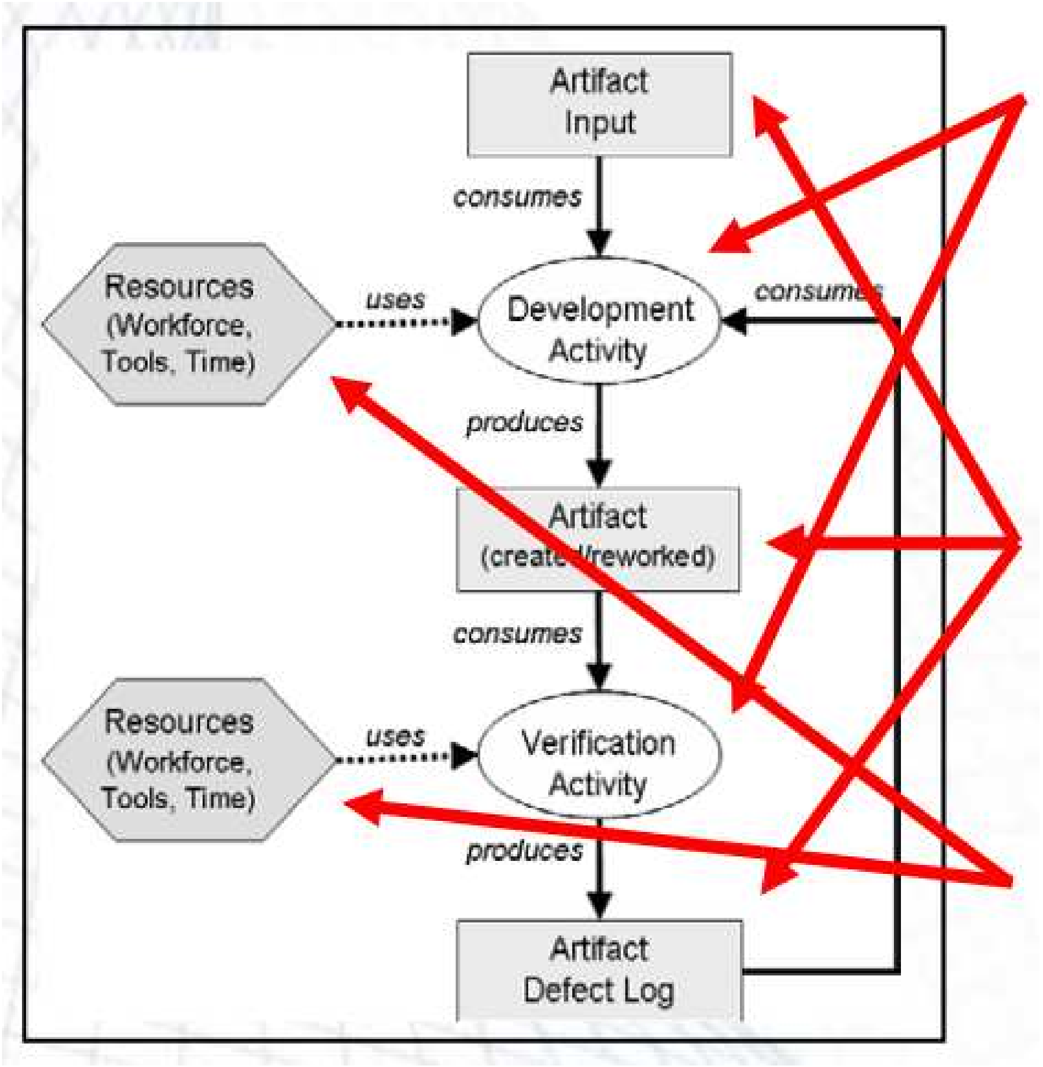
$$Object = \left\{ \begin{array}{ll} attribute_1 & (value_{11}, value_{12}, \ldots) \\ attribute_2 & (value_{21}, value_{22}, \ldots) \\ \ldots & \ldots \\ attribute_n & (value_{n1}, value_{n2}, \ldots) \end{array} \right\}$$

>> **Metrics** are standards (i.e. commonly accepted scales) that define measureable attributes of entities, their units and their scopes.

>> **Measure** is a relation between an attribute and a measurement scale.

What is Measurement?

>> An **entity** in software measurement can be any of the following:



What is Measurement?

>> An **attribute** is a feature or property of an **entity**

- E.g., blood pressure of a person, cost of a journey, duration of a software specification process.

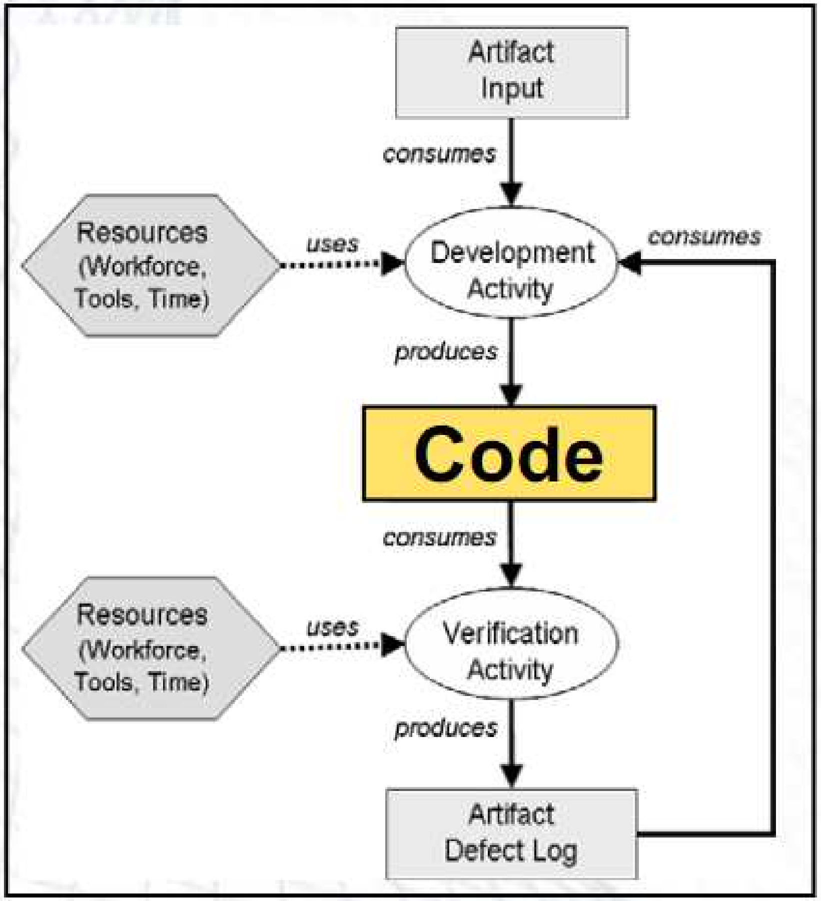
>> There are two general types of attributes: *Internal attribute* and *External attributes*.

>> **Internal attribute** of an entity can be measured only based on the entity itself.

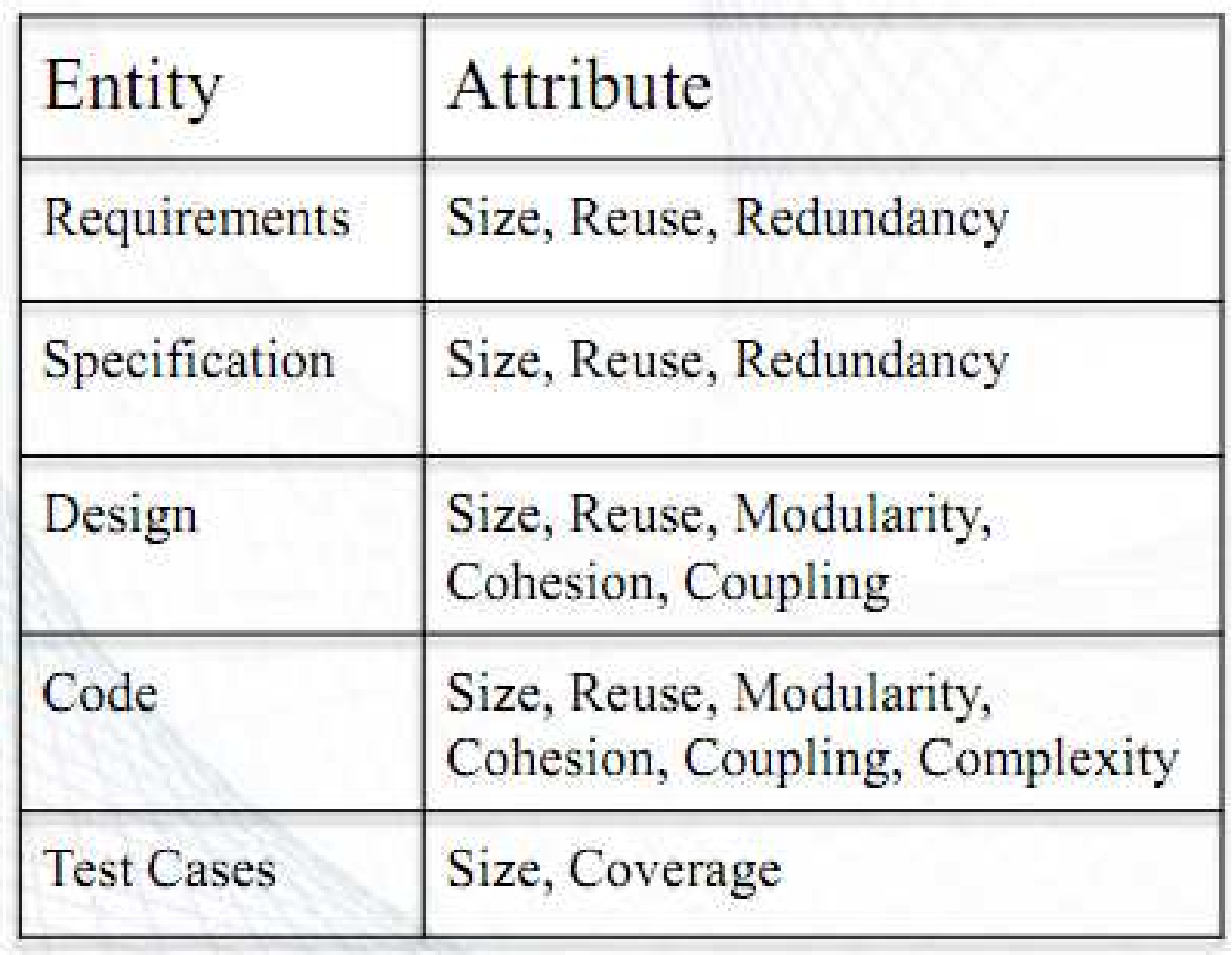
- e.g., **entity**: code, **internal attribute**: size, modularity, coupling.

>> **External attributes** of an entity can be measured only with respect to how the entity relate to its environment

- e.g., **entity**: code, **external attribute**: reliability, maintainability.



What is Measurement?



Entity	Attribute
Requirements	Size, Reuse, Redundancy
Specification	Size, Reuse, Redundancy
Design	Size, Reuse, Modularity, Cohesion, Coupling
Code	Size, Reuse, Modularity, Cohesion, Coupling, Complexity
Test Cases	Size, Coverage

Measurement Example

Measurement: Types

>> Measurements are needed as:

→ **Descriptors** of entities already in existence

→**Prescriptors** (standards, norms, failure intensity objectives, benchmarks) which entities of certain class or category should satisfy.

→**Predictors** to estimate properties of entities yet to be designed or implemented

Measurement: How to

>> In order to make entities measurable:

→What entities (objects) should be selected?
→ What attributes should be selected?
→ What values should be assigned to the attributes?
→ What shall be rules (relationship) ascribed to the attributes and their entities?

Note: assigned values and/or ascribed rules can be either quantitative or qualitative

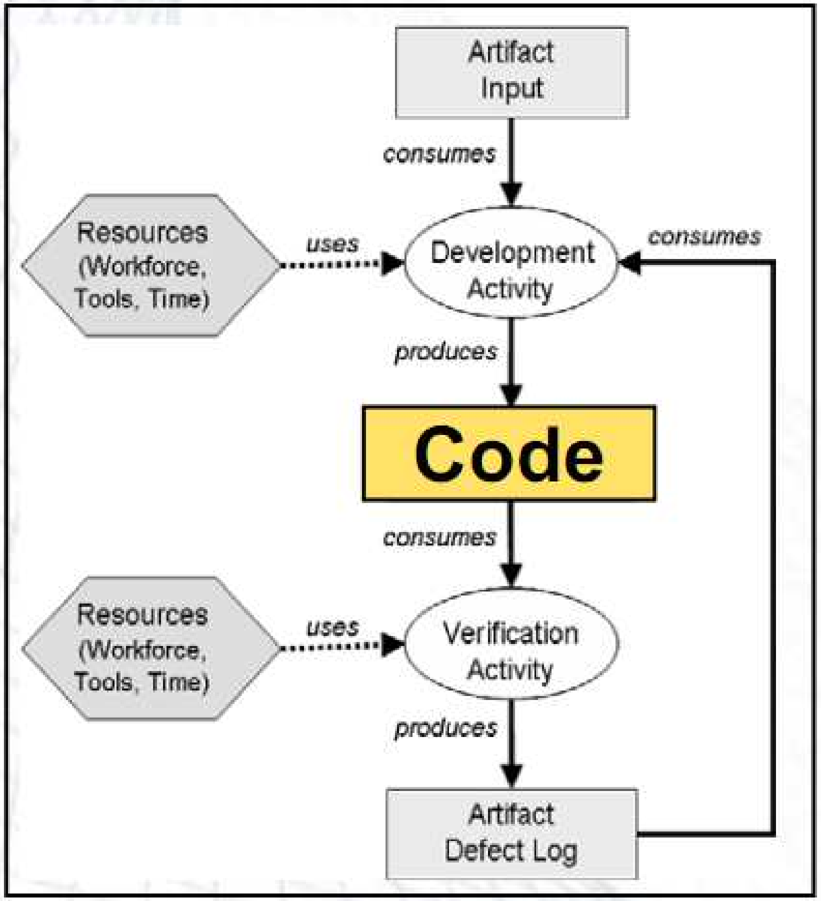
Measurement: Example 1

>> Entity: Code

>> Attribute: Size

>> Possible Measures:

- NCSLOC (Not Commented Source Lines of Code)
- # Statement
- # Modules
- # Procedures
- # Classes
- # Methods
- etc….



Measurement: Example 2

>> Entity: Availability

>> Attribute: System Up-time, Down-time

>> Values: Time in Seconds

>> Relations: Availability = Up-time / (Up-time + Down-time)

Software Metrics Challenges

>> Measuring Physical Entities:

Entity	Attribute	Unit	Value
Human	Height	cm	178

>> Measuring Non-Physical Entities:

Entity	Attribute	Unit	Value
Human	IQ	IQ Index	89

>> Software Engineering (SE) Metrics are mostly Non-Physical

- Reliability, Maturity, Portability, Flexibility, Maintainability, etc. and Relations are unknown

What is Software Measurement?

>> Software metrics are measures that are used to quantify software, software development resources and/or software development process.

>> This includes items which are directly measureable, such as ***lines of code (LOC)***, as well as which are calculated from measurements, such as ***software quality***.

Measurement in SE

>> Measurement in SE is selecting, measuring and putting together may different attributes of the software, and adding our subjective interpretations in order to get a whole picture of the software.

>> This a not a trivial task!

>> 300+ metrics have been defined.

Measurement in SE

>> Before a measurement project can be planned

- Objective and scope should be established
- Alternative solutions should be considered
- Technical and management constraints should be identified.

>> This information is required to estimate costs, project tasks, and a project schedule.

Measurement in SE

>> In order to manage software measurement project one must understand and plan:

- The goal and scope of work
- Risks
- Resources required
- Tasks to be accomplished
- Milestones to tracked
- Total costs of the project
- Schedule to be followed

Scope of Software Metrics

→Cost and effort estimation

→ Productivity measures and models

→ Data collection

→ Quality models and measures

→ Reliability models

→ Performance evaluation and models

→ Structural and complexity metrics

→ Capability and maturity assessment

→ Management by metrics

→ Evaluation of methods and tools

Scope of Software Metrics

>> Cost and effort estimation

- Software cost estimation is the process of predicting the amount of effort required to build a software system.

- Estimates for project cost and time requirements are derived during the planning stage of a project.

-Constructive Cost Model (COCOMO) is one of the model used to estimate cost.

- Models provide mathematical algorithms to compute cost as a function of a number of variables such as size (using lines of code, function points, etc.) and/or complexity (using cyclomatic complexity, etc.).

- Most of the models are available as automated tools.

Summary

>> Without measurement there is no way to determine if the process/product are improving.

>> Metrics allow the establishment of meaningful foals for improvement. A baseline from which improvements can be measured can be established.

>> Metrics allow us to identify the causes of defects which have major effect on software development.

>> When metrics are applied to a product they help identify:

- Which user requirements are likely to change
- Which modules are most error prone
- How much testing should be planned for each module

References

→**Software Metrics**

B.H. Far

Department of Electrical & Computer Engineering

University of Calgary