PRODUCT, PROCESS AND PROJECT METRICS

Lecture # 39

Lecture by Engr. Sidra

Objectives

- After completing this chapter you will:
  - Understand why measurement and metrics are important in software engineering
  - What is difference between product, process and project metrics
  - What guidelines, attributes and principles are required to be followed to create good metrics
  - Know about different metrics from each category

Software Metrics

- **Product metrics** – Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
- **Process metrics** – These characteristics can be used to improve the development and maintenance activities of the software.
- **Project metrics** – This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.
  - Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.

Why Do We Measure?

- Product metrics provide a basis from which analysis, design, coding, and testing can be conducted more objectively and assessed more quantitatively.
- Assess the status of an ongoing project
- Track potential risks
- Uncover problem areas before they go "critical,"
- Adjust workflow or tasks,
- Evaluate the project team's ability to control quality of software work products.

Measures, Metrics and Indicators

- A *measure* provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process
- The IEEE glossary defines a *metric* as "a quantitative measure of the degree to which a system, component, or process possesses a given attribute."
- An *indicator* is a metric or combination of metrics that provide insight into the software process, a software project, or the product itself

Measurement Principles

- The objectives of measurement should be established before data collection begins;
- Each technical metric should be defined in an unambiguous manner;
- Metrics should be derived based on a theory that is valid for the domain of application (e.g., metrics for design should draw upon basic design concepts and principles and attempt to provide an indication of the presence of an attribute that is deemed desirable);
- Metrics should be tailored to best accommodate specific products and processes [Bas84]

Measurement Process

- *Formulation.* The derivation of software measures and metrics appropriate for the representation of the software that is being considered.
- *Collection*. The mechanism used to accumulate data required to derive the formulated metrics.
- *Analysis*. The computation of metrics and the application of mathematical tools.
- *Interpretation.* The evaluation of metrics results in an effort to gain insight into the quality of the representation.
- *Feedback.* Recommendations derived from the interpretation of product metrics transmitted to the software team.

Goal-Oriented Software Measurement

- The Goal/Question/Metric Paradigm
  1. Establish an explicit measurement *goal* that is specific to the process activity or product characteristic that is to be assessed
  2. Define a set of *questions* that must be answered in order to achieve the goal, and
  3. Identify well-formulated *metrics* that help to answer these questions.
- Goal definition template
  - Analyze {the name of activity or attribute to be measured}
  - for the purpose of {the overall objective of the analysis}
  - with respect to {the aspect of the activity or attribute that is considered}
  - from the viewpoint of {the people who have an interest in the measurement}
  - in the context of {the environment in which the measurement takes place}.

Metrics Attributes

- *Simple and computable*. It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time
- *Empirically and intuitively persuasive.* The metric should satisfy the engineer's intuitive notions about the product attribute under consideration
- *Consistent and objective.* The metric should always yield results that are unambiguous.
- *Consistent in its use of units and dimensions.* The mathematical computation of the metric should use measures that do not lead to bizarre combinations of unit.
- *Programming language independent.* Metrics should be based on the analysis model, the design model, or the structure of the program itself.
- *Effective mechanism for quality feedback.* That is, the metric should provide a software engineer with information that can lead to a higher quality end product