Overview

- Software process and project metrics are quantitative measures that enable software engineers to gain insight into the efficiency of the software process and the projects conducted using the process framework. In software project management, we are primarily concerned with productivity and quality metrics. The four reasons for measuring software processes, products, and resources (to characterize, to evaluate, to predict, and to improve).

Measures, Metrics, and Indicators

- Measure - provides a quantitative indication of the size of some product or process attribute
- Measurement - is the act of obtaining a measure
- Metric - is a quantitative measure of the degree to which a system, component, or process possesses a given attribute

Process and Project Indicators

- Metrics should be collected so that process and product indicators can be ascertained
- Process indicators enable software project managers to: assess project status, track potential risks, detect problem area early, adjust workflow or tasks, and evaluate team ability to control product quality

Process Metrics (Refer Fig: 4.1)

- Private process metrics (e.g. defect rates by individual or module) are known only to the individual or team concerned.
- Public process metrics enable organizations to make strategic changes to improve the software process.
- Metrics should not be used to evaluate the performance of individuals.
- Statistical software process improvement helps and organization to discover where they are strong and where are week.

Project Metrics

- Software project metrics are used by the software team to adapt project workflow and technical activities.
- Project metrics are used to avoid development schedule delays, to mitigate potential risks, and to assess product quality on an on-going basis.
- Every project should measure its inputs (resources), outputs (deliverables), and results (effectiveness of deliverables).

Software Measurement (Refer Section 4.3 Completely)

- Direct measures of software engineering process include cost and effort.
- Direct measures of the product include lines of code (LOC), execution speed, memory size, defects per reporting time period.
- Indirect measures examine the quality of the software product itself (e.g. functionality, complexity, efficiency, reliability, maintainability).

Size-Oriented Metrics

- Derived by normalizing (dividing) any direct measure (e.g. defects or human effort) associated with the product or project by LOC.
- Size oriented metrics are widely used but their validity and applicability is widely debated.

Function-Oriented Metrics

- Function points are computed from direct measures of the information domain of a business software application and assessment of its complexity.
- Once computed function points are used like LOC to normalize measures for software productivity, quality, and other attributes.
- Feature points and 3D function points provide a means of extending the function point concept to allow its use with real-time and other engineering applications.
- The relationship of LOC and function points depends on the language used to implement the software.

Software Quality Metrics

- Factors assessing software quality come from three distinct points of view (product operation, product revision, product modification).
- Software quality factors requiring measures include correctness (defects per KLOC), maintainability (mean time to change), integrity (threat and security), and usability (easy to learn, easy to use, productivity increase, user attitude).
- Defect removal efficiency (DRE) is a measure of the filtering ability of the quality assurance and control activities as they are applied through out the process framework. (Refer section 4.5.3)

Integrating Metrics with Software Process

- Many software developers do not collect measures.
- Without measurement it is impossible to determine whether a process is improving or not.
- Baseline metrics data should be collected from a large, representative sampling of past software projects.
- Getting this historic project data is very difficult, if the previous developers did not collect data in an on-going manner.

Statistical Process Control (Refer Fig. 4.7)

- It is important to determine whether the metrics collected are statistically valid and not the result of noise in the data.
- Control charts provide a means for determining whether changes in the metrics data are meaningful or not.
- Zone rules identify conditions that indicate out of control processes (expressed as distance from mean in standard deviation units).

Metrics for Small Organizations

- Most software organizations have fewer than 20 software engineers.
- Best advice is to choose simple metrics that provide value to the organization and don’t require a lot of effort to collect.
- Even small groups can expect a significant return on the investment required to collect metrics, if this activity leads to process improvement.

Establishing a Software Metrics Program

- Identify business goal
- Identify what you want to know
- Identify sub goals
- Identify sub goal entities and attributes
- Formalize measurement goals
- Identify quantifiable questions and indicators related to sub goals
- Identify data elements needed to be collected to construct the indicators

- Define measures to be used and create operational definitions for them
- Identify actions needed to implement the measures
- Prepare a plan to implement the measures

REFER TO CHAPTER 4 OF ***"Pressman. R. S., Software Engineering a practitioners Approach. 5th Edition"*** ACCORDINGLY.