CHAPTER 22 PROCESS AND PROJECT METRICS

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. There are four reasons for measuring software processes, products, and resources (to characterize, to evaluate, to predict, and to improve).

Measures and Metrics

- **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 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

- Private process metrics (e.g., defect rates by individual or module) are only known to by 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 they are weak.

Statistical Process Control

1. Errors are categorized by their origin
2. Record cost to correct each error and defect
3. Count number of errors and defects in each category
4. Overall cost of errors and defects computed for each category
5. Identify category with greatest cost to organization
6. Develop plans to eliminate the most costly class of errors and defects or at least reduce their frequency
Project Metrics

- A software team can use software project metrics 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

- Direct measures of a software engineering process include cost and effort.
- Direct measures of the product include lines of code (LOC), execution speed, memory size, defects reported over some time period.
- Indirect product 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 a matter of some debate.

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.
- The relationship of LOC and function points depends on the language used to implement the software.

Object-Oriented Metrics

- Number of scenario scripts (NSS)
- Number of key classes (NKC)
- Number of support classes (e.g., UI classes, database access classes, computations classes, etc.)
- Average number of support classes per key class
- Number of subsystems (NSUB)

Web Engineering Project Metrics

- Number of static Web pages (Nsp)
- Number of dynamic Web pages (Nd p)
- Customization index: \( C = \frac{Nsp}{(Ndp + Nsp)} \)
- Number of internal page links
- Number of persistent data objects
- Number of external systems interfaced
- Number of static content objects
- Number of dynamic content objects
- Number of executable functions

### 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)
  - 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 throughout the process framework
  \[ DRE = \frac{E}{E + D} \]
  \( E = \) number of errors found before delivery of work product
  \( D = \) number of defects found after work product delivery

### 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.

### 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

1. Identify business goal
2. Identify what you want to know
3. Identify subgoals
4. Identify subgoal entities and attributes
5. Formalize measurement goals
6. Identify quantifiable questions and indicators related to subgoals
7. Identify data elements needed to be collected to construct the indicators
8. Define measures to be used and create operational definitions for them
9. Identify actions needed to implement the measures
10. Prepare a plan to implement the measures