25.1.2 Project Metrics

Process metrics that are used for strategic purposes, software project measures are calculated. Project metrics used by a project manager and a software team to adapt project workflow and technical activities. Project metric is used to estimate effort and time. The project manager uses these data to monitor and control progress.

Other project metrics:-

Production rates represented in terms of models created, review hours, function points, and delivered source lines are measured.

Technical metrics are collected to assess design quality and to provide indicators that will influence the approach taken to code generation and testing.

1. These metrics are used to minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks.
2. Project metrics are used to assess product quality on an ongoing basis and, when necessary, modify the technical approach to improve quality.

As quality improves, defects are minimized, and as the defect count goes down, the amount of rework required during the project is also reduced. This leads to a reduction in overall project cost.

25.2 SOFTWARE MEASUREMENT

Measurements in the physical world can be categorized in two ways: direct measures (e.g., the length of a bolt) and indirect measures (e.g., the "quality" of bolts produced, measured by counting rejects). Software metrics can be categorized similarly.

Direct measures of the software process include cost and effort applied. Direct measures of the product include lines of code (LOC) produced, execution speed, memory size, and defects reported over some set period of time.

Indirect measures of the product include functionality, quality, complexity, efficiency, reliability, maintainability.

Partitioned the software metrics domain into process, project, and product metrics and noted that product metrics that are private to an individual are often combined to develop project metrics that are public to a software team.

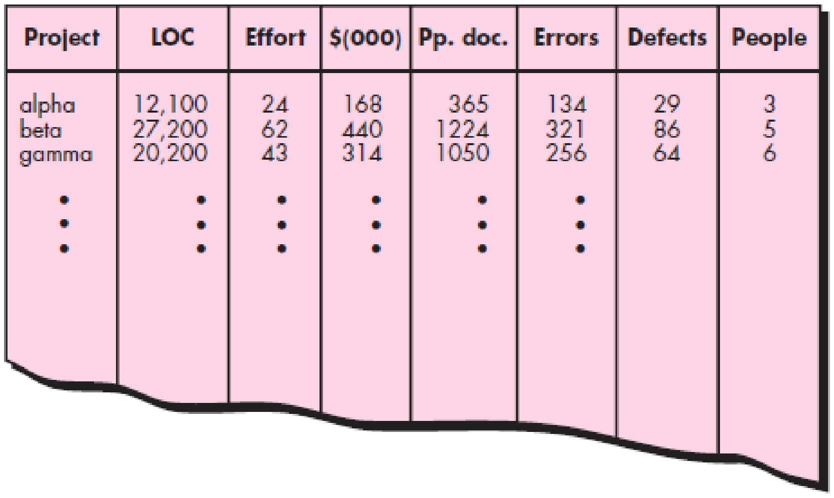
Project metrics are then consolidated to create process metrics that are public to the software organization as a whole.

25.2.1 Size-Oriented Metrics

Size-oriented software metrics are derived by normalizing quality and/or productivity measures by considering the size of the software that has been produced. If a software organization maintains simple records, a table of size-oriented measures, such as the one shown in Figure 25.2, can be created. The table lists each software development project that has been completed over the past few years and corresponding measures for that project. Referring to the table entry (Figure 25.2) for project alpha: 12,100 lines of code were developed with 24 person-months of effort at a cost of $168,000.

FIGURE 25.2

Size-oriented metrics



Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
⋮	⋮	⋮	⋮	⋮	⋮		

Three people worked on the development of software for project alpha. In order to develop metrics that can be assimilated with similar metrics from other projects, you can choose lines of code as a normalization value. From the rudimentary data contained in the table, a set of simple size-oriented metrics can be developed for each project:

- Errors per KLOC (thousand lines of code)
- Defects per KLOC
- $ per KLOC
- Pages of documentation per KLOC

In addition, other interesting metrics can be computed:

- Errors per person-month

• KLOC per person-month

• $ per page of documentation

Size-oriented metrics are not universally accepted as the best way to measure the software process.

25.2.2 Function-Oriented Metrics

Function-oriented software metrics use a measure of the functionality delivered by the application as a normalization value. The most widely used function-oriented metric is the function point (FP). Computation of the function point is based on characteristics of the software's information domain and complexity.

It is also controversial like LOC. FP is independent of programming language, making it ideal for applications using conventional and nonprocedural languages. It is based on data that are more likely to be known early in the evolution of a project, making FP more attractive as an estimation approach.

25.2.3 Reconciling LOC and FP Metrics

The relationship between lines of code and function points depends upon the programming language that is used to implement the software and the quality of the design.

- One LOC of C++ provides approximately 2.4 times the "functionality" (on average) as one LOC of C.
- One LOC of Smalltalk provides at least four times the functionality of an LOC for a conventional programming language such as Ada, COBOL, or C.

LOC and FP measures are often used to derive productivity metrics. This invariably leads to a debate about the use of such data.

a) Should the LOC/person-month (or FP/person-month) of one group be compared to similar data from another?
b) Should managers appraise the performance of individuals by using these metrics?

Answer is NO. The reason for this response is that many factors influence productivity.

Function points and LOC-based metrics have been found to be relatively accurate predictors of software development effort and cost. You should be concerned primarily with productivity and quality-Measures of software development "output" as a function of effort and time, and measures of the "fitness for use" of the work products that are produced.

Reference:- Roger S Pressman - Software engineering _ a practitioner's approach-McGraw-Hill Higher Education (2010)