Unit-II
Measures, Metrics and Indicators

Measure: The Quantitative indication of the extent, amount, dimension, or size of some attribute of a product or process. A single data point.

Metrics: The degree to which a system, component, or process possesses a given attribute. Relates several measures (e.g. average number of errors found per person hour).

Indicators: A combination of metrics that provides insight into the software process, project or product.

Metrics in the Process and Project Domains: -
The only rational way to improve any process is

- To measure specific attributes of the process.
- Develop a set of meaningful metrics based on these attributes.
- Use the metrics to provide indicators that will lead to a strategy for improvement.

Processes is the center of a triangle connecting three factors that have profound influence on software quality and organizational performance shown in fig:-2.1.

- The skill and motivation of people has most influential factor in quality and performance.
- The complexity of the product has impact on quality and team performance.
- The technology (the software engineering methods) the process triangle exists within a circle of environmental conditions that include the development environment, business conditions, customer characteristics.

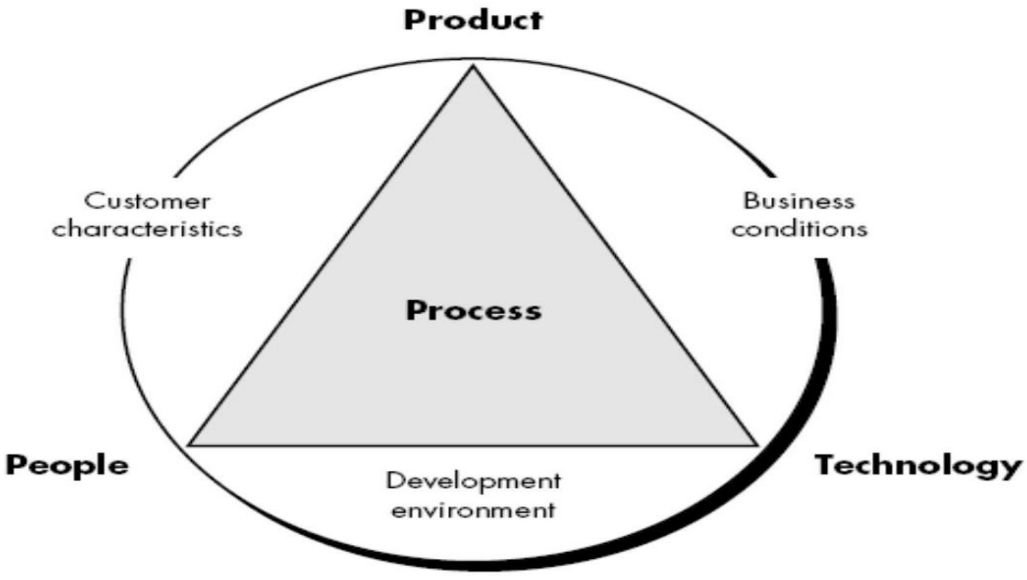

Fig:-2.1

Software Measurement
Software measurement is a quantified attribute of a characteristic of a software product or the software process.
Measurement helps in estimation, quality control, productivity assessment and project control throughout a software project.

Software measurements are of two categories
1. Direct measures
2. Indirect measures.

1. Direct measures:- It include software processes like cost and effort applied and products like lines of code produced, execution speed, and other defects that have been reported.
2. Indirect measures:- It include products like functionality, quality, complexity, reliability, maintainability, and many more.

Measurement process is characterized by a set of five activities, they are :-

- Formulation: This performs measurement and develops appropriate metric for software under consideration.
- Collection: This collects data to derive the formulated metrics.
- Analysis: This calculates metrics and the use of mathematical tools.
- Interpretation: This analyzes the metrics to attain insight into the quality of representation.
- Feedback: This communicates recommendation derived from product metrics to the software team.

SOFTWARE PRODUCT METRICS AND SOFTWARE PROCESS METRICS: -

Software process metrics measure the software development process and environment. Example productivity, effort estimates, efficiency and failure rate.
Software Product metrics measure the software product.
Example: - size, reliability, complexity and functionality.

Metrics of Software Quality

Software metrics can be classified into three categories −

- 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.

Software quality metrics can be further divided into three categories −

1. Product quality metrics
2. In-process quality metrics
3. Maintenance quality metrics

1. Product Quality Metrics

This metrics include the following −

- Mean Time to Failure
  It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

- Defect Density
  It measures the defects relative to the software size expressed as lines of code or function point, etc.

i.e., it measures code quality per unit. This metric is used in many commercial software systems.

- Customer Problems
  It measures the problems that customers encounter when using the product. It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.
- Customer Satisfaction
  Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys.

2. In-process Quality Metrics

In-process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes −

- Defect density during machine testing
  Defect rate during formal machine testing (testing after code is integrated into the system) is correlated with the defect rate in the field. Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.

- Defect arrival pattern during machine testing
  The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field.

- Phase-based defect removal pattern
  A large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software.

3. Maintenance Quality Metrics

Fix quality or the number of defective fixes is another important quality metric for the maintenance phase. A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction. The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.

Software Reliability

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability.

Software estimation techniques

The technique which is used to calculate the time required to accomplish a particular task is called Estimation Techniques. To estimate a task different effective Software Estimation Techniques can be used to get the better estimation. It is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable. Estimation determines how much money, effort, resources, and time it will take to build a specific system or product. Estimation is based on −

- Past Data/Past Experience
- Available Documents/Knowledge
- Assumptions

- Identified Risks

The four basic steps in Software Project Estimation are −

- Estimate the size of the development product.
- Estimate the effort in person-months or person-hours.
- Estimate the schedule in calendar months.
- Estimate the project cost in agreed currency.

LOC and FP estimation

Lines of code (LOC) and function points (FP) were described as measures from which productivity metrics can be computed. LOC and FP data are used in two ways during software project estimation:
(1) As an estimation variable to "size" each element of the software and
(2) As baseline metrics collected from past projects ` and used in conjunction with estimation variables to develop cost and effort ` projections.
LOC and FP estimation are distinct estimation techniques.
The problems of lines of code (LOC):-
– Different languages lead to different lengths of code.
– It is not clear how to count lines of code.
– A report, screen, or GUI generator can generate thousands of lines of code in minutes.
– Depending on the application application, the complexity complexity of code is different.

The Five Components of Function Points

1. Internal Logical Files
2. External Interface Files
3. External Inputs
4. External Outputs
5. External Inquiries

1. Internal Logical Files - The first data function allows users to utilize data they are responsible for maintaining.

2. External Interface Files - The second Data Function a system provides an end user is also related to logical groupings of data. In this case the user is not responsible for maintaining the data. The data resides in another system and is maintained by another user or system. The user of the system being counted requires this data for reference purposes only.

3. External Input - The first Transactional Function allows a user to maintain Internal Logical Files (ILFs) through the ability to add, change and delete the data.

4. External Output - The next Transactional Function gives the user the ability to produce outputs.

5. External Inquiries - The final capability provided to users through a computerized system addresses the requirement to select and display specific data from files. To accomplish this user inputs selection information that is used to retrieve data that meets the specific criteria. In this situation there is no manipulation of the data. It is a direct retrieval of information contained on the files.

Empirical models - COCOMO

The COCOMO is defined as COnstructive COst Model.It is Based on water fall process model

There are three forms of the COCOMO

- Basic COCOMO (macro estimation) which gives an initial rough estimate of man months and development time.
- Intermediate COCOMO which gives a more detailed detailed estimate estimate for small to medium sized projects.
- Detailed COCOMO (micro estimation) which gives a more detailed estimate for large projects.

REVERSE ENGINEERING

It is a process to achieve system specification by thoroughly analyzing, understanding the existing system. This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analyzing lower abstraction levels.

An existing system is previously implemented design, about which we know nothing. Designers then do reverse engineering by looking at the code and try to get the design. With design in hand, they try to conclude the specifications. Thus, going in reverse from code to system specification.