Software Metrics

Software metric

- A software metric is a measure of software characteristics which are measurable or countable.
- Software metrics are valuable for measuring
  - software performance,
  - planning work items,
  - measuring productivity,
  - and many other uses

Need of software Matrics

- Create the quality of the current product or process.
- Anticipate future qualities of the product or process.
- Enhance the quality of a product or process.
- Regulate the state of the project in relation to budget and schedule.

Steps related to software matrics

A metrics is a measurement of the level that any impute belongs to a system product or process. There are 4 functions related to software metrics:

- Planning
- Organizing
- Controlling
- Improving

Classification of Software Metrics

1. **Product Metrics:** These are the measures of various characteristics of the software product. The two important software characteristics are:

I. Size and complexity of software.

II. Quality and reliability of software.

2. Process Metrics: These are the measures of various characteristics of the software development process. They are used to measure the characteristics of methods, techniques, and tools that are used for developing software

1. the efficiency of fault detection.
2. the effectiveness of defect removal during development,
3. the pattern of testing defect arrival
4. the response time of the fix process.

3. Project metrics : It describe the project characteristics and execution. As:

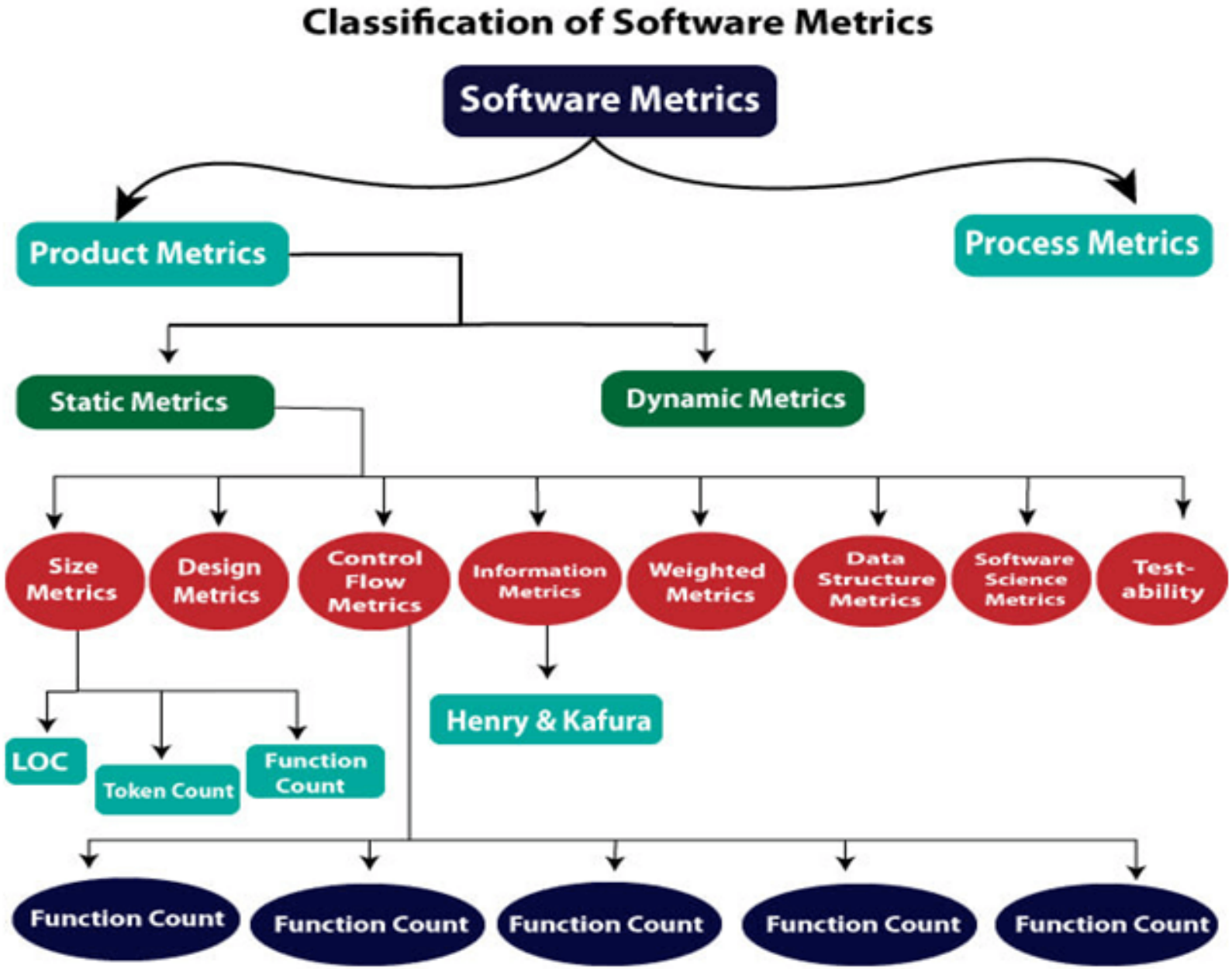
1. include the number of software developers,
2. the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

Scope of Software Metrics

- Cost and effort estimation
- Productivity measures and model
- Data collection
- Quantity models and measures
- Reliability models
- Performance and evaluation models
- Structural and complexity metrics
- Capability – maturity assessment
- Management by metrics
- Evaluation of methods and tools

Characteristics of software Metrics:

- **Quantitative:**
  Metrics must possess quantitative nature.It means metrics can be expressed in values.
- **Understandable:**
  Metric computation should be easily understood ,the method of computing metric should be clearly defined.
- **Applicability:**
  Metrics should be applicable in the initial phases of development of the software.
- **Repeatable:**
  The metric values should be same when measured repeatedly and consistent in nature.
- **Economical:**
  Computation of metric should be economical.
- **Language Independent:**
  Metrics should not depend on any programming language.


Classification of Software Metrics
Software Metrics
Product Metrics
Process Metrics
Static Metrics
Dynamic Metrics
Size Metrics
Design Metrics
Control Flow Metrics
Information Metrics
Weighted Metrics
Data Structure Metrics
Software Science Metrics
Test-ability
LOC
Token Count
Function Count
Henry & Kafura
Function Count
Function Count
Function Count
Function Count
Function Count

Types of Metrics

- **Internal metrics:** Internal metrics are the metrics used for measuring properties that are viewed to be of greater importance to a software developer. For example, Lines of Code (LOC) measure.
- **External metrics:** External metrics are the metrics used for measuring properties that are viewed to be of greater importance to the user, e.g., portability, reliability, functionality, usability, etc.
- **Hybrid metrics:** Hybrid metrics are the metrics that combine product, process, and resource metrics. For example, cost per FP where FP stands for Function Point Metric.
- **Project metrics:** Project metrics are the metrics used by the project manager to check the project's progress. Data from the past projects are used to collect various metrics, like time and cost; these estimates are used as a base of new software. Note that as the project proceeds, the project manager will check its progress from time-to-time and will compare the effort, cost, and time with the original effort, cost and time. Also understand that these metrics are used to decrease the development costs, time efforts and risks. The project quality can also be improved. As quality improves, the number of errors and time, as well as cost required, is also reduced.

Cocomo (Constructive Cost Model)

- It is a regression model based on LOC, i.e **number of Lines of Code**.

1. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule

- **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Different system types

- **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
- **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.
- **Embedded –** A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models

Types of Models:

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements. These are types of COCOMO model:

- Basic COCOMO Model
- Intermediate COCOMO Model
- Detailed COCOMO Model

About...

- **Basic COCOMO** can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.
- **Intermediate Model –**The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software system. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation
- **Detailed Model –**
  Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.