

Software Metrics

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

Within the software development process, many metrics are that are all connected. Software metrics are similar to the four functions of management: Planning, Organization, Control, or Improvement.

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.

Advantage of Software Metrics

Comparative study of various design methodology of software systems.

For analysis, comparison, and critical study of different programming language concerning their characteristics.

In comparing and evaluating the capabilities and productivity of people involved in software development.

In the preparation of software quality specifications.

In the verification of compliance of software systems requirements and specifications.

In making inference about the effort to be put in the design and development of the software systems.

In getting an idea about the complexity of the code.

In taking decisions regarding further division of a complex module is to be done or not.

In guiding resource manager for their proper utilization.

In comparison and making design tradeoffs between software development and maintenance cost.

In providing feedback to software managers about the progress and quality during various phases of the software development life cycle.

In the allocation of testing resources for testing the code.

Disadvantage of Software Metrics

The application of software metrics is not always easy, and in some cases, it is difficult and costly.

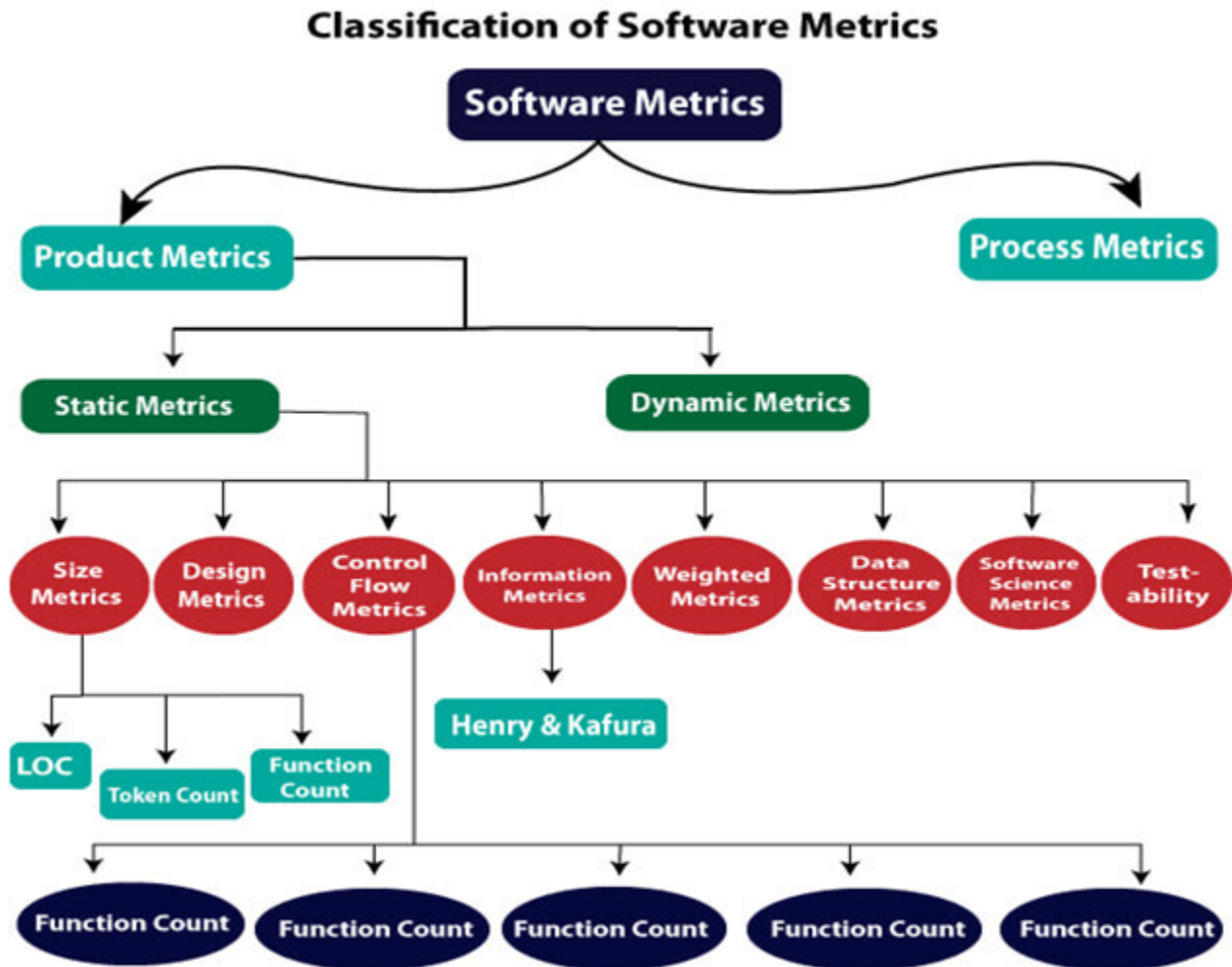
The verification and justification of software metrics are based on historical/empirical data whose validity is difficult to verify.

These are useful for managing software products but not for evaluating the performance of the technical staff.

The definition and derivation of Software metrics are usually based on assuming which are not standardized and may depend upon tools available and working environment.

Most of the predictive models rely on estimates of certain variables which are often not known precisely.

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



Project Metrics :

The processes which comes under the software development project needs to be measured in order to get assured the well progress of software development.

Software Metrics provide measures for various aspects of software process and software product.

There are various type of Metrics like:

- a) **Process metrics**
- b) **project metrics**
- c) **product metrics**
- d) **organizational metrics**

What is a Project Metrics:

These are metric that pertain to project quality. They are used to quantify defects, cost, schedule, productivity and estimation of various resources and deliverables.

- **Schedule variance:** Any difference between the schedule completion of an activity the actual completion of an activity and the actual completion is known as schedule variance.

$$\text{Schedule variance} = \frac{((\text{Actual Calendar days} - \text{Planned Calendar days}) + \text{Start Variance})}{\text{Planned Calendar day}} \times 100$$

Effort Variance

Schedule variance = ((actual calendar days - planned calendar days) + start variance) / planned calendar day * 100

- **Effort variance (EV) :** Difference between the planned outlined efforts and the effort required to actually undertake the task is called effort variance.

Effort variance metric is called Effort Variance

$$EV = \frac{(\text{Actual Effort} - \text{Planned Effort})}{\text{Planned Effort}} \times 100$$

EV = (actual effort - planned effort) / planned effort * 100

- **Size Variance:** Difference between the estimated size of the project and the actual size of the project (normally KLOC or FP)

$$\text{Size Variance} = \frac{(\text{Actual Size} - \text{Estimated Size})}{\text{Estimated Size}} \times 100$$

Size variance = (actual size - estimated size) / estimated size 100

- **Requirement Stability Index:** provides visibility to the magnitude and impact of requirement changes

$$\text{RSI} = 1 - \frac{(\text{No. of changed} + \text{No. of deleted} + \text{No. of added})}{\text{Total No. of Initial Requirements}} \times 100$$

RSI = 1 - ((number of changed + number of deleted + number of added) / total number of initial requirements) * 100

- **Productivity (Project):** It is a measure of output from a related process for a unit of input.

Project productivity = actual project size / actual effort expended in the project.

$$\text{Project Productivity} = \frac{\text{Actual Project Size}}{\text{Actual effort expended in the project.}}$$

- **Productivity (For test case preparation):** = actual number of test cases / actual effort expended in the test case preparation.

- **Productivity (For test case execution):** = actual number of test cases / actual effort expended in testing.

- **Productivity (defect detection):** = actual number of defects (review + testing) / actual effort spent on (review + testing)

● **Productivity (defect fixation):** = actual number of defect fixed/ actual effort spent on defect fixation.

● **Schedule variance for a phase:** The deviation between plan and actual schedules for the phase within a project.

= **actual calendar days for a phase- planned calendar days for a phase + start variance for a phase)/(planned calendar days for a phase)*100**

● **Effort variance for a phase :**=(actual effort for a phase- planned effort for a phase)/(planned effort for a phase)*100

Project Size Estimation Metrics

Four metrics are popularly being used to estimate size

a) count the lines

b) lines of code(LOC)

c) function point(FP)

d) feature Point

1) count the lines:

There are several ways to count the lines in the code. Depending on what you count, you get a low or a high line count.

Count the line supports

● -> **Physical Lines:** Bitmetric count the all physical lines(LINES)

● -> **logical lines:** It covers one or more physical lines. Two or more physical lines can be joined as one logical line with the line continuation sequence(LLINES)

● -> **logical lines of code:** A logical lines of code is one that contains actual source code. An empty line or a comment line is not counted in LOC.

Physical lines of code is not supported by count the LINES. This type of a metric count the lines and comments.

2) lines of code(LOC)

LOC measure the size of a project by counting the number of source instructions in the development program ignoring the commenting code and header lines.

Determining the LOC at beginning is very difficult than in the end. Problem is divided into modules and each module into sub modules and so on until the size of the different Leaf-Level modules can be approximately predicted for estimating and loc in beginning.

Shortcomings of LOC

1) LOC gives a numerical value of problem size

Problem-

If may vary widely with individual coding style.

Example- one programmer may write several instructions in a single line but another program may write these instructions in several lines.

Solution-

Count the language tokens rather than the lines of code.

2) LOC is a measure of the coding activity alone.

Problem:

Good program size measure should consider the total effort for specify, design, code, test etc.

3) - It merely computes the number of sources line in the final program

Problem:

Coding is a very small part in the overall software development activities.

4) LOC metrics measure the lexical Complexity of a program and does not address the more important issue of logical or structure complexity.

Problem:

program having Complex logic would require much more effort to develop than a program with simple logic.

LOC measures correlates poorly with the quality and efficiency of the code.

Problem:

Large code size does not imply better quality or higher efficiency.

Key points:

- the LOC count can only be accurately completed only after the has been fully developed .

- LOC metric penalizes use of high level programming reuse etc.

- If a programmer use many library routines or he/she reuse code then the LOC of that problem is less but it does not mean that the effort of that program is very few.

So it is not right way to estimate the project size.

Function Point Metric(FPM)

This metric overcomes the shortcoming of the LOC metric

Why we use it? Because it can be used to easily estimate the size of the software product directly from the problem specification. The conceptual Idea behind the FPM

is that size of the software product is directly dependent on the number of different functions or features it supports.

Steps to compute function point:

- When function is invoked, read some input data and transform it to corresponding output data.

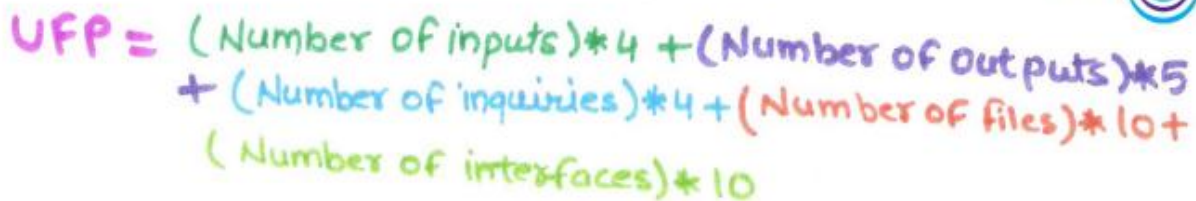
Example the issue book feature of a library automation software takes the name of the book as input and displays its location and the number of copies available.

- Beside using the number of the input and output data values function point metric computes the size of a software product **(in unit of function points or FPs.)**

Function point is computed in two steps:

- 1) computing the unadjusted function point(UFP):** UFP is refined to reflect the differences in the complexity of the different parameters.

UFP=(number of input)*4+(number of outputs)*5+(number of enquiries)*4+(number of files)*10+ (number of interfaces)*10



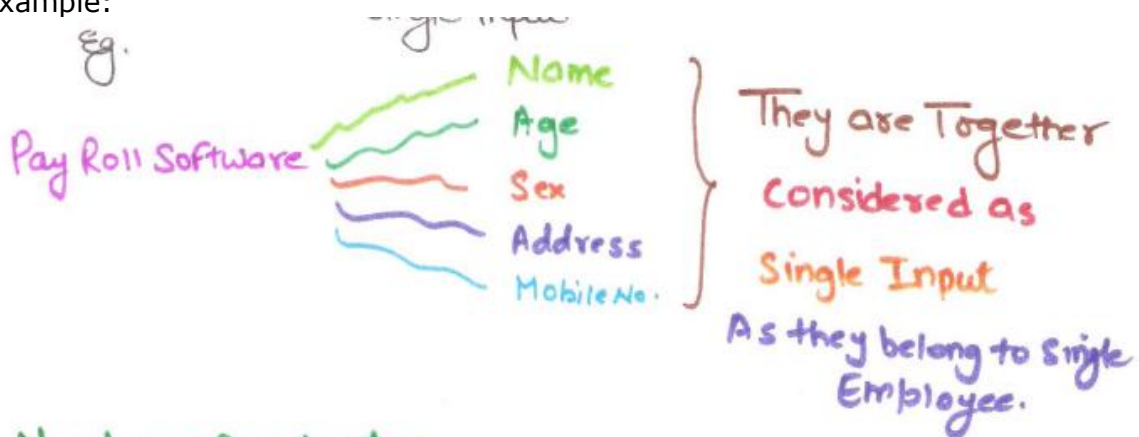
A handwritten formula for UFP is shown in a light blue box. The formula is: $UFP = (\text{Number of inputs}) * 4 + (\text{Number of outputs}) * 5 + (\text{Number of inquiries}) * 4 + (\text{Number of files}) * 10 + (\text{Number of interfaces}) * 10$. The terms are color-coded: 'Number of inputs' is green, 'Number of outputs' is purple, 'Number of inquiries' is blue, 'Number of files' is red, and 'Number of interfaces' is yellow. There is a small circular logo in the top right corner of the box.

- 1) number of inputs:** In this is data item input by the user is counted. Data import should be distinguished from the inquiries.



Individual data items input by the user are not considered in the calculation of the number of input, but a group of related inputs are considered as single input.

Example:



Number of outputs:

2) **Number of outputs:** it refers to- reports printed, screen outputs, error messages produced



While outputting the number of output individual data items with in a reports are not considered, but a set of related data items is counted as one input.

3) number of inquiries: Distinct interactive queries which can be made by the users. These inquiries are the user commands which require specified action by the system.

4) Number of files: Each logical file is counted.



5) Number of interface: Interfaces used to exchange information with other systems.



Once the unadjusted function point is computed, the technical complexity factor(TCF) is computed next.

TCF refines the UFP measured by considering 14 other factors such as high transaction rate, throughput, and response time requirement etc.

Each of these 14 factor is assigned from 0(not present or no influence) to 6 (strong influence). The resulting number are summed, yielding the total degree of influence (DI).

Now TCF is computed as

$$=(0.65+0.1*DI)$$

and DI vary from 0-84 and TCF Vary from (0.65-1.35)

so $FP=UFP*TCF$

Shortcomings of Function Point Metric:

- **Subjective Evaluations:** It needs subjective evaluation with a lot of judgement involved.
- **Conversion need:** Many efforts and models are based on LOC, a function point need to be converted.
- **Less Researched Data:** Less research data is available on function point as compared to LOC.
- **Late performance:** It is performed after creation of design specification.
- **Low Accuracy:** It has low accuracy of evaluating as a subjective judgement is involved.
- **Long learning curve:** As the learning curve is quite long it's not easy to gain proficiency.
- **Time consuming:** It is a time consuming method as less research data is available which generate low accuracy and less effective results.

4) Feature Point Metric:

A function point extension called feature points, is a superset of the function point measure that can be applied to systems and Engineering software applications. The feature point measure accommodate applications in which **Algorithm Complexity is high.**

To compute the Feature Point:

Information domain values are again counted and weighted. In addition, another software characteristics- algorithm counted.



Project Estimation Techniques

The main activities which comes under software project planning:

- # Estimation,
- # Scheduling,
- # Risk Analysis,
- # Quality Management Planning,
- # Change management planning.

Estimation: It Is an attempt to determine how much money, efforts, resources and time it will take to build a specific software based system or project.

Who does estimation? :

- Software manager does estimation using information collected from customers and software Engineers and
- software metrics data collected from past projects.
- Assumption taken from experience .

- Identified risks and feasibility helps in estimation.

Steps for Estimations:

- -> Estimate the size of the development product
- -> Estimate the effort in person-month or person - hours



- -> Estimate the schedule in calendar months.
- -> Estimate the project cost in agreed currency.

Before a final estimate is made, problem complexity and risk are considered.

"**Estimate Risk**" is measured by the degree of uncertainty in the quantitative estimate established by the resources, and schedule.