Software metrics in software engineering are the standards for estimating the quality, **progress**, and health of **software development** activity. A software metric is a quantifiable or countable assessment of the qualities of software.

**Characteristics of Software Metrics**

Software metrics in software engineering should possess the following characteristics:

* **Quantitative**: Metrics must be quantitative in order to be helpful. It means that metrics can be **stated numerically**.
* **Understandable**: Metric computation should be simple to **grasp**, and the method for **computing** them should be well explained.
* **Applicability**: Metrics should be applied during the early stages of **software development**.
* **Repeatable**: When measured regularly and **consistently in nature**, the metric values should be the same.
* **Economical**: Metric computation should be **cost-effective**.
* **Language** **agnostic**: Metrics should be **language-independent**, meaning their computation should not depend on any **programming language**.

**Need for Software Metrics**

Software metrics in software engineering can be used for a variety of purposes, including analyzing software performance, planning work items, estimating productivity, and so on. For example, you can use software metrics to **monitor performance**, plan upcoming work tasks, **track productivity**, and better regulate the production process throughout project management if you can view distinct statistics and trends as production takes place. You can also use software metrics in conjunction with management functions to simplify your projects by building more **efficient procedures**, creating software maintenance plans, and keeping production teams updated on issues that need to be resolved.

Software metrics are needed for various reasons, including the following:

* By preventing faults, it is possible to save money.
* Aids in the **improvement** of overall project planning
* Aids in the **advancement** of overall project planning
* Encourages **eagerness** to improve processes
* Aids in a more thorough analysis of the **dangers involved**.
* Aids in studying **indicators** at each testing stage to increase problem eradication efficiency.
* Increases the **ROI** of test automation over time
* Improves the relationship between **testing coverage**, **risks**, and **system complexities**.

**Classification of Software Metrics**

There are three types of software metrics in software engineering: **product metrics**, **process metrics**, and **project metrics**. Product metrics are used to define product attributes such as size, **design features**, **complexity**, performance, **level of quality**, and so on. In contrast, process metrics are used to optimize software development and maintenance. Project metrics, such as the number of software developers, **cost**, and so on, indicate the project's characteristics and perfor

**Product Metrics**

Product metrics represent product attributes such as size, **complexity**, **design features**, **performance**, and **quality level**. A working product is created at the end of each successive phase of the **software development process**. At any step of development, a product can be measured. Metrics are built for these items to determine whether a product is being developed in accordance with user requirements. If a product fails to satisfy **consumer expectations**, the relevant steps are made in the appropriate phase. Product metrics assist software engineers in detecting and correcting possible issues before they cause catastrophic failures.

* In software development process, a working product is developed at the end of each successful phase. Each product can be measured at any stage of its development.
* Metrics are developed for these products so that they can indicate whether a product is developed according to the user requirements. If a product does not meet user requirements, then the necessary actions are taken in the respective phase.
* Product metrics help software engineer to detect and correct potential problems before they result in catastrophic defects. In addition, product metrics assess the internal product attributes in order to know the efficiency of the following.
* Analysis, design, and code model
* Potency of test cases
* Overall quality of the software under development.

Various metrics formulated for products in the development process are listed below.

* **Metrics for analysis model:** These address various aspects of the analysis model such as system functionality, system size, and so on.
* **Metrics for design model:** These allow software engineers to assess the quality of design and include architectural design metrics, component-level design metrics, and so on.
* **Metrics for source code:** These assess source code complexity, maintainability, and other characteristics.
* **Metrics for testing:** These help to design efficient and effective test cases and also evaluate the effectiveness of testing.
* **Metrics for maintenance:** These assess the stability of the software product.

## ****Metrics for the Analysis Model****

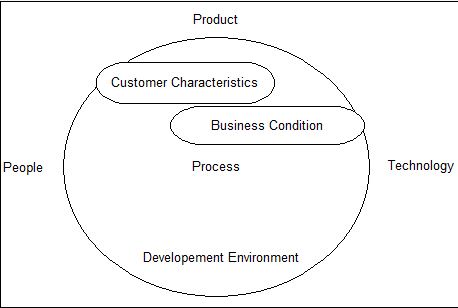
There are only a few metrics that have been proposed for the analysis model. However, it is possible to use metrics for project estimation in the context of the analysis model. These metrics are used to examine the analysis model with the objective of predicting the size of the resultant system. Size acts as an indicator of increased coding, integration, and testing effort; sometimes it also acts as an indicator of complexity involved in the software design. Function point and lines of code are the commonly used methods for size estimation.

### Function Point (FP) Metric

**Process Metrics**

To improve any process, it is important to measure its defined properties, establish a set of meaningful metrics based on these attributes, and then utilize these metrics to generate indicators to develop a process improvement strategy. Software engineers can measure the performance of a software process that is executed using the process as a framework by utilizing **software process metrics**. The process is at the center of the triangle that connects three aspects **(product, people, and technology)** that significantly impact software quality and organizational performance. People's competence and motivation, the complexity of the product, and the level of technology utilized in software development all significantly **impact quality** and **team performance**.

* To improve any process, it is necessary to measure its specified attributes, develop a set of meaningful metrics based on these attributes, and then use these metrics to obtain indicators in order to derive a strategy for process improvement.
* Using software process metrics, software engineers are able to assess the efficiency of the software process that is performed using the process as a framework.
* Process is placed at the centre of the triangle connecting three factors (product, people, and technology), which have an important influence on software quality and organization performance.
* The skill and motivation of the people, the complexity of the product and the level of technology used in the software development have an important influence on the quality and team performance.
* The process triangle exists within the circle of environmental conditions, which includes development environment, business conditions, and customer /user characteristics.



To measure the efficiency and effectiveness of the software process, a set of metrics is formulated based on the outcomes derived from the process. These outcomes are listed below.

* Number of errors found before the software release
* Defect detected and reported by the user after delivery of the software
* Time spent in fixing errors
* Work products delivered
* Human effort used
* Time expended
* Conformity to schedule
* Wait time
* Number of contract modifications
* Estimated cost compared to actual cost.

process metrics are of two types, namely, private and public. Private Metrics are private to the individual and serve as an indicator only for the specified individual(s). Defect rates by a software module and defect errors by an individual are examples of private process metric

**Project Metrics**

Project metrics explain the characteristics and execution of a project. Examples include the number of software developers, the staffing pattern throughout the **software's life cycle**, **cost**, **schedule**, and **productivity**. The project manager monitors the project's progress using measures known as project metrics. Various metrics, such as time, cost, and so on, are collected using data from previous projects and utilised as an estimate for the new initiative. The project manager monitors the project's progress on a **regular basis**, and **effort**, **time**, and cost are compared to the initial effort, time, and cost. These indicators can help lower development costs, efforts, hazards, and time. The project's quality can also be improved. With an improvement in quality, there is a decrease in the number of errors, time, cost, and so on.

* Project metrics enable the project managers to assess current projects, track potential risks, identify problem areas, adjust workflow, and evaluate the project team’s ability to control the quality of work products.
* Project metrics serve two purposes.
* One, they help to minimize the development schedule by making necessary adjustments in order to avoid delays and alleviate potential risks and problems.
* Two, these metrics are used to assess the product quality on a regular basis-and modify the technical issues if required.
* As the quality of the project improves, the number of errors and defects are reduced, which in turn leads to a decrease in the overall cost of a software project.

**Types of Software Metrics**

* **Internal metrics:** Internal metrics are used to measure properties that are deemed more important to a software developer than the users. Lines of Code **(LOC)** is one example.
* **External metrics:** External metrics are used to measure features that are deemed more important to the user than the software developers, such as **portability**, **reliability**, **functionality**, **usability**, and so on.
* **Hybrid Metrics:** Metrics that **mix product**, **process**, and resource metrics are known as **hybrid metrics**. Cost per FP is an example, where FP stands for Function Point Metric.

**Advantages of Software Metrics**

* The software system design technique can be analyzed in comparison to other designs.
* Using software metrics, the properties of various **programming languages** can be explored for study and comparison.
* Software metrics in **software engineering** can be used to create software quality requirements.
* Software system compliance with **requirements** and standards can be verified.
* It is possible to infer the effort required to **design** and **develop** software systems.
* The code's **complexity** can be calculated.
* It is possible to decide whether or not to divide a **complex module** into smaller ones.
* It is possible to direct the best use of **resource managers**.
* It is possible to make design **trade-offs** and compare maintenance and software development expenses.

**Disadvantages of Software Metrics**

* It isn't easy to apply measurements in all situations. In some **circumstances**, it is difficult and costly.
* It is challenging to validate historical or empirical evidence used for **verification** and **argumentation**.
* Although software products can be managed, technical staff performance cannot be measured using **software metrics**.
* The available tools and working environment are utilized to define and **derive software metrics**, and there is no standard for doing so.
* Certain variables are estimated using **predictive models** and are not always known.