Part 1: Software Measurements and Metrics

Software metrics deals with the measurement of the software product and the process by which it is developed. The software product should be viewed as an abstract object that evolves from an initial statement of need to a finished software system, including source and object code and the various forms of documentation produced during development. These measurements of the software process and product are studied and developed for use in modeling the software development process. These metrics and models are then used to estimate/predict product costs and schedules and to measure productivity and product quality. Information gained from the metrics and the model can then be used in the management and control of the development process, leading, to an improved result.

So, Software is measured to:

- Indicate the quality of the product.
- Assess the productivity of the people who produce the product.
- Assess the benefits derived from new software engineering tools and methods.
- Form a baseline for estimation.
- Help justify requests for new tools or training.

Good metrics should facilitate the development of models that are capable of predicting process or product parameters, not just describing them.
So, ideal metrics should be:

- Simple, precisely definable, so that it is clear how the metric can be evaluated.
- Objective, to the greatest extent possible.
- Easily obtainable i.e., at reasonable cost.
- Valid, i.e. the metric should measure what it is intended to measure.
- Robust, i.e. relatively insensitive to insignificant changes in the process or product. [4]

Measurement Scales for Software Metrics:

Before collecting the software metric data, it is important to consider the type of information involved. Four basic types of measured data are recognized by statisticians—nominal, ordinal, interval, and ratio. The four basic types of data are described by the following table: [4]

Type of Data	*Possible Operations*	*Description of Data*
Nominal	=, ≠	Categories
Ordinal	<, >	Rankings
Interval	+, -	Differences
Ratio	/	Absolute zero

1. **Nominal:** e.g. no ordering, simply attachment of labels (language: 3GL, 4GL).
2. **Ordinal:** e.g. ordering, but no quantitative comparison (Programmer capability: low, average, high).
3. **Interval:** e.g. between certain values (programmer capability: between 55^{th} and 75^{th} percentile of the population ability).
4. **Ratio:** e.g. (the proposed software is twice as big as the software that has just been completed).

1.a: Measurement:

- Effective management of the software development process requires effective measurement of that process. [1]
- Software measurement is concerned with deriving a numeric value for an attribute of a software product or process. [4]

Classification of Software Measurements:

Measurements can be either *direct* or *indirect*:

- Direct measures, are taken from a feature of an item. They are calculated by Size-oriented metrics which are a direct measure of software and the process by which it was developed. It measures an attribute directly.
- Indirect measures associate a measure to a feature of the object being measured. It measure an attribute based on other. They are calculated by function-oriented measurements.

The following table gives example of both direct and indirect measurements:

Direct	Indirect
LOC	Functionality
Speed	Productivity
Memory size	Effectiveness
Defects reported	Quality
Cost	Complexity
Effort(time)	Reliability
Documentation pages	Maintainability
Duration of test process	
People in project	

From the measurements above a list of both simple size-oriented metrics and function –oriented metrics can be generated for example:

Simple size-oriented	Function-oriented
▪ Productivity = LOC / person-hour ▪ Quality = defects / LOC ▪ Cost = Cost / LOC ▪ Documentation = pages of documentation / LOC	▪ Programmer productivity (LOC/hour). ▪ Module defect density (defects/LOC). ▪ Defect detection efficiency (# defects per time/total defects). ▪ Requirements stability:(Initial requirements/total requirements). ▪ Test effectiveness (enumerable covered/total enumerable). ▪ System spoilage (debugging effort/total effort).

The following table is a summery table of the Metric list derived from the information above:

Category	Metric
Productivity	LOC/person-hour
	Story/person-hour
	Release/month
Speed	LOC/hour
	Story/hour
	Release/hour
quality	Defects/LOC
	Defects/Story
	Defects/Release
Cost	Cost/LOC
	Cost/Day
Documentation	Pages of documentation/LOC
Efficiency	# of defects per time/total defects
Requirements stability	Initial requirements/actual requirements

1.b: Metrics:

- Software metrics are numerical data related to software development. Metrics strongly support software project management activities. [1]
- Software metrics deals with the measurement of the software product and the process by which it is developed. [4]

Classification of Software Metrics:

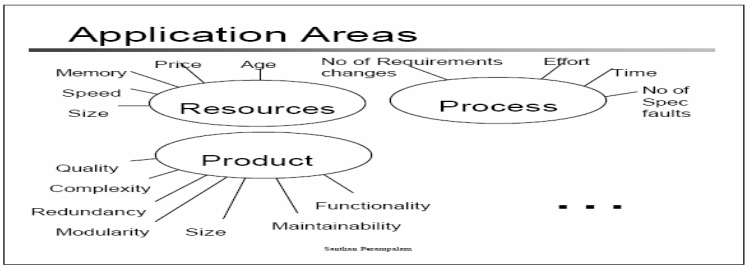
Metrics can be categorized in several ways, the most common ways are:

A) The first method of categorization is as 'Product, Project or Process'

- **Product metrics**, measure the internal or external attributes of the developed software product itself, at any stage of its development. [3]
  For Example:
  - Size Metrics: There are three types of size metrics:
    1. Lines of Code.
    2. Function points count.
    3. Bang.
  - Complexity Metrics: there are four types of complexity metrics:
    1. Cyclomatic Complexity—v(G).
    2. Extensions to v(G).
    3. Knots.
    4. Information Flow.
  - Program vocabulary length (N) metric
  - Volume (V) metric
  - Quality Metrics
    1. Defect Metrics
    2. Reliability Metrics
    3. Maintainability Metrics
- **Process metrics** are measures of the software development process, in order to optimize the software development process itself. They are collections of software-related activities like: [4]
  - Application of methods and tools.
  - Use of standards.
  - Effectiveness of management.
  - Performance of development system.

  It is also possible to use product metrics calculated on the process description. [4]
- **Project metrics**, as the name implies, focuses on a specific development project. Such as : [7]
  - Percentage of test cases passed
  - Number of defects found by integration testing

Some other resources considered only *Product* and *Process* metrics, and some others included *Recourses* metrics as the image shown below:



- And **Resource metrics** are defined as entities required by a process activity. Examples of resource metrics are:

  - Effort expended (On tasks within a project, classified by lifecycle phase software function, On extra-project activities training)
  - Elapsed time
  - Computer resources. [4]

B) Another way is as 'Primitive or Computed'

- **Primitive metrics** are those that can be directly observed, such as

  - Program size (in LOC)
  - Number of defects observed in unit testing
  - Total development time for the project.

- **Computed metrics** are those that cannot be directly observed but are computed in some manner from other metrics. Examples of computed metrics are those commonly used for productivity, such as

  - (LOC/person-month)
  - Number of defects per thousand lines of code (defects/KLOC).

  Computed metrics are combinations of other metric values and thus are often more valuable in understanding or evaluating the software process than are simple metrics.[4]

Software Metrics and Measurements available:

Category	Metrics Description t	Example
Progress	Provides information on how well the project is performing with respect to its schedule.	Actual vs. planned task completions Actual vs. planned durations [1] LOC/ Hour , Story/ Hour , Release/ Hour [5]
Effort	Provides visibility into the contributions of staffing on project costs, schedule adherence, and product quality.	Actual vs. planned staffing profiles [1] LOC/person-hour [6]
Cost	Provides tracking of actual costs against estimated costs and predicts future costs.	Actual vs. planned costs Cost and schedule variances [1] Cost/ LOC , Cost/ Day [5]
Review Results	Provides status of (*AI*) Action Items from life-cycle review.	Status of action items
Trouble Reports	Provides insight into product and process quality and the effectiveness of the testing.	Status of trouble reports Number of trouble reports opened, closed, etc. During reporting period [1] Defects/LOC Number of defects per time/ Total defects [6]
Requirements Stability	Provides visibility into the magnitude and impact of requirements changes.	Number of requirements changes/clarifications Distribution of requirements over releases
Size Stability	Provides insight into the completeness and stability of the requirements and into the ability of the staff to complete the project within the current budget and schedule.	Size growth Distribution of size over releases
Computer Resource Utilization	Provides information on how well the project is meeting its computer resource utilization goals/requirements.	Actual vs. planned profiles of computer resource utilization
Training	Provides information on the training program and staff skills.	Actual vs. planned number of personnel attending training classes [1]
Complexity	Measures program complexity.	Block of sequential code and its path to other code (it's calculated by mathematic formula) [3]
Design Metrics	Analyzes design information in order to develop metrics available during architectural and detailed design that help software developers identify error-prone modules in a software design.	Software Systems/ Architecture [2] Pseudo code Measures Communication/Interaction Measures Object Oriented Design Measures Review/Inspection/Audits Measures

Code Metrics	Metrics related to the coding	Source Code Measures, Lines of code (LOC) Data (Flow) Measures Object Oriented Programming Functional Programming Logical Programming Visual Programming
Test Metrics	Metrics related to the testing	Test Coverage/ OO Testing Reliability Measurement Security Measurement Performance Measurement
Maintenance Metrics	Provide information about the modification measurements a software need	Out Modifiability/Portability Measures Reusability/COTS Measures Programmers Productivity Measures Evaluation/Certification Measures
Requirement Analysis Metrics	Provide information related to the requirements meeting and the risk management.	Project Requirement/Risk Management Problem Definition Text Analysis Requirement Analysis

Metrics included in our Projects:

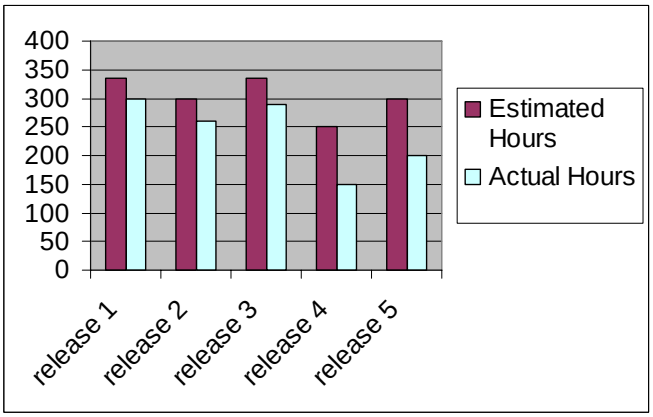
We will include in out project two types of process metrics: Progress and Effort.

1- Progress Metrics (Project Level)

It provides information on how well the project is performing with respect to planned time. So the comparison between Actual vs. planned Estimate Time for all Project releases.

Ex:

Release	Estimated Hours	Actual Hours
release 1	336	300
release 2	300	260
release 3	336	290
release 4	250	150
release 5	298	200



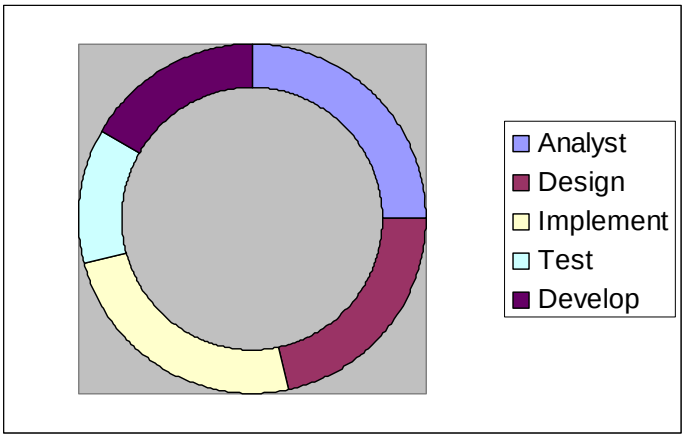
2- Effort Metrics (Story Level)

It provides information on amount of effort required for each story. So the comparison between Story tasks actual time with other Story tasks.

Ex:

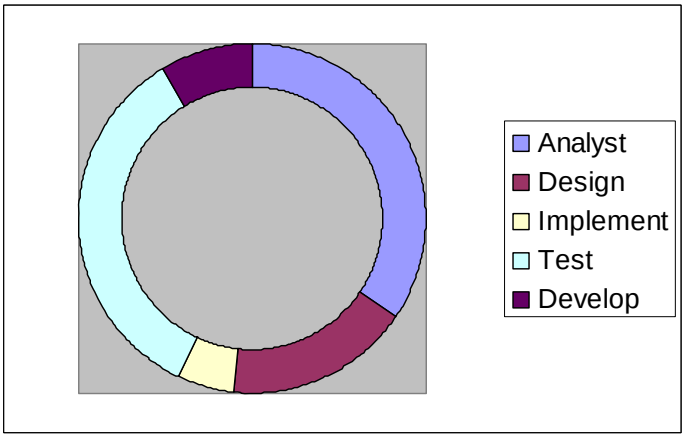
Story #1

Tasks	Actual Hours
Analysis	300
Design	260
Implementation	290
Testing	150
Development	200



Story #2

Tasks	Actual Hours
Analysis	200
Design	100
Implementation	30
Testing	200
Development	50



Metrics rejected from our Projects:

All the other types of software metrics require extra programming and extra skills, and due to the short period of time that we have to completely finish our project, we focused on two types of process metrics witch we mentioned above.

Rejected Metrics	Comment	Reason for reject
Cost	This Metrics allows managers to track the actual costs and measure them against the budget for their respective areas of responsibility.	Our project doesn't calculate the budget, Team members work for free also resources are free. There is no cost and no budget
Review Results	This Metrics record the AIs (*Action Item*) identified in the review findings and track them until they are resolved	Our project doesn't support Tracking Processing
Trouble Reports	This Metrics applicable only in (1) application test and integration, (2) system test, (3) acceptance test.	Our project deals with testing just in application form to be filled
Requirements / Size Stability	This Metrics indicate the completeness, stability, and understanding of the requirements	Our project assumes stabile requirements
Computer Resource Utilization	This Metrics concern in computer resources i.e. CPU time, I/O, and memory	Out of Our Project scope
Training	This Metrics provide managers with information on the training program and whether the staff has necessary skills	Our project staff made Self-Training
Complexity	There are many type of complexity, and each is calculated using a special formula	Need so complicated Mathematic calculation
Design Metrics **Code Metrics** **Test Metrics** **Maintenance Metrics** **Requirement Analysis Metrics.. etc.**	These types of metrics provide several designs, coding, testing maintenance and checking the requirements product.	Out of project scope