Metrics for Process and Projects

What is Software metrics?

Software metrics (process and project) are quantitative measures. Measurement can be applied:

To the software process with the intent of improvement

To assist in estimation, quality control, productivity assessment, and project control

To help assess the quality of technical work products

To assist in tactical decision making as a project proceeds

Why we need to measure?

To characterize in an effort to gain an understanding of process, products, resources and environments and to establish baselines for comparisons with future assessments.

To evaluate to determine status with respect to plans

To predict by gaining understanding of relationships among processes and products and building models of these relationships

To improve by identifying roadblocks, root causes, inefficiencies, and other opportunities for improving product and process performance.

Metric and Indicator

A **metric** is a quantitative measure of the degree to which a system, component, or process possesses a given attribute.

An **indicator** is a metric or combination of metrics that provide insight into the software process, a software project, or the product itself.

A software engineer collects measures and develops metrics so that indicators will be obtained .

Process Metrics

Process metrics are collected across all projects and over long periods of time.

Process metrics provide a set of process indicators that lead to long-term software process improvement

The only way to know how/where 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

Project metrics

Enable a software project manager to

- Assess the status of an ongoing project
- Track potential risks
- Uncover problem areas before they "go critical“
- Adjust work flow or tasks

Project metrics

Evaluate the project team's ability to control quality of software engineering work products

Estimate effort and time duration

Every project should measure input, output and result

Use of Project Metrics

The first application of project metrics occurs during estimation

- Metrics from past projects are used as a basis for estimating time and effort

As a project proceeds, the amount of time and effort expended are compared to original estimates

As technical work commences, other project metrics become important

- Production rates are measured (represented in terms of models created, review hours, function points, and delivered source lines of code)
- Error uncovered during each generic framework activity (i.e, communication, planning, modeling, construction, deployment) are measured

Use of Project Metrics

Project metrics are used to

- Minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks
- Assess product quality on an ongoing basis and, when necessary, to modify the technical approach to improve quality

In summary

- As <u>quality improves</u>, defects are minimized
- As <u>defects go down</u>, the amount of rework required during the project is also reduced
- As <u>rework goes down</u>, the overall project <u>cost is reduced</u>

Summary: Project Metrics

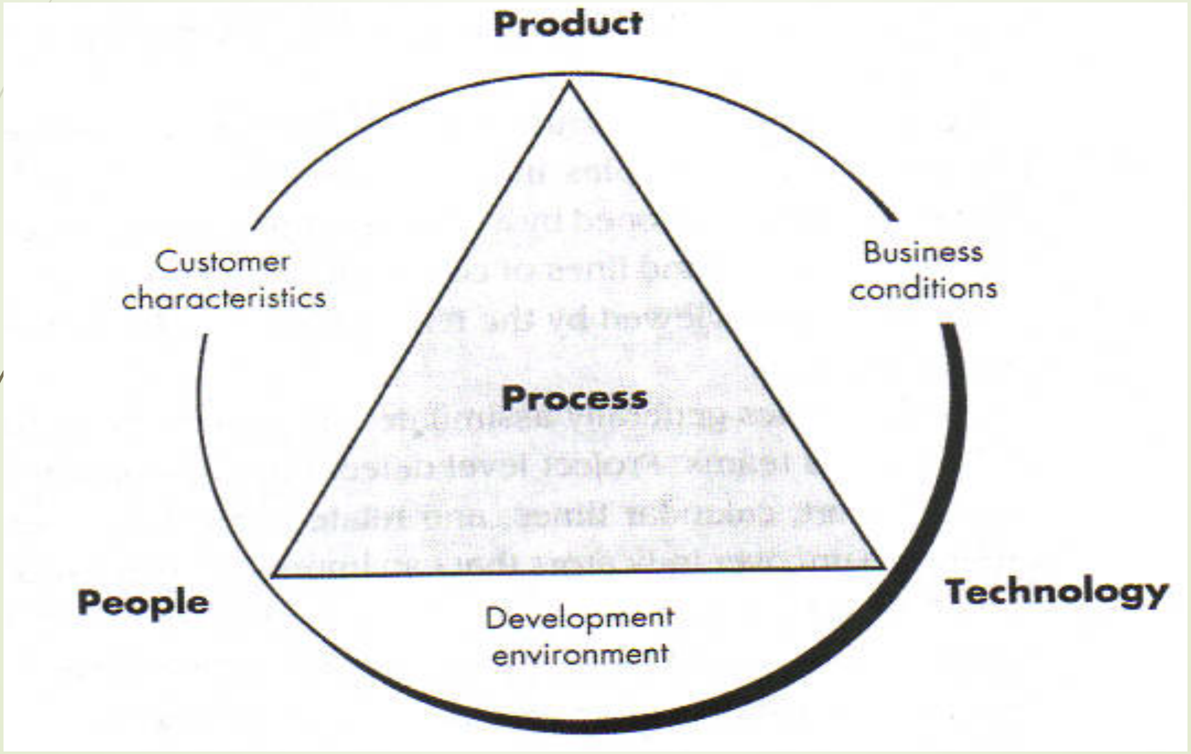
Used by project manager and a software team to adapt project workflow and technical activities.

Metrics collected from past projects used to estimate effort and time.

Project metrics measures: review hours, function points, delivered source lines, errors uncovered during each software

Intent of project metrics is twofold: 1. these metrics are used to minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks.

Determinants for Software quality & Organizational Effectiveness



Private and Public metric

There are "private and public" uses for different types of process data

Data *private* to the individual

? Serve as an indicator for the individual only

Eg : Defect rates, Errors found during development

Public metric

Defects reported for major software functions

Errors found during formal technical reviews

Lines of code or function points per module/function

Software Measurement

Direct measures

- Software process (cost)
- Software product (lines of code (LOC), execution speed, defects reported over some set period of time

Indirect measures

- Software product (Functionality, quality, complexity, efficiency, reliability, maintainability)

Many factors affect software work, it is difficult (don’t use metrics) to compare individuals/team

Size-Oriented Metrics

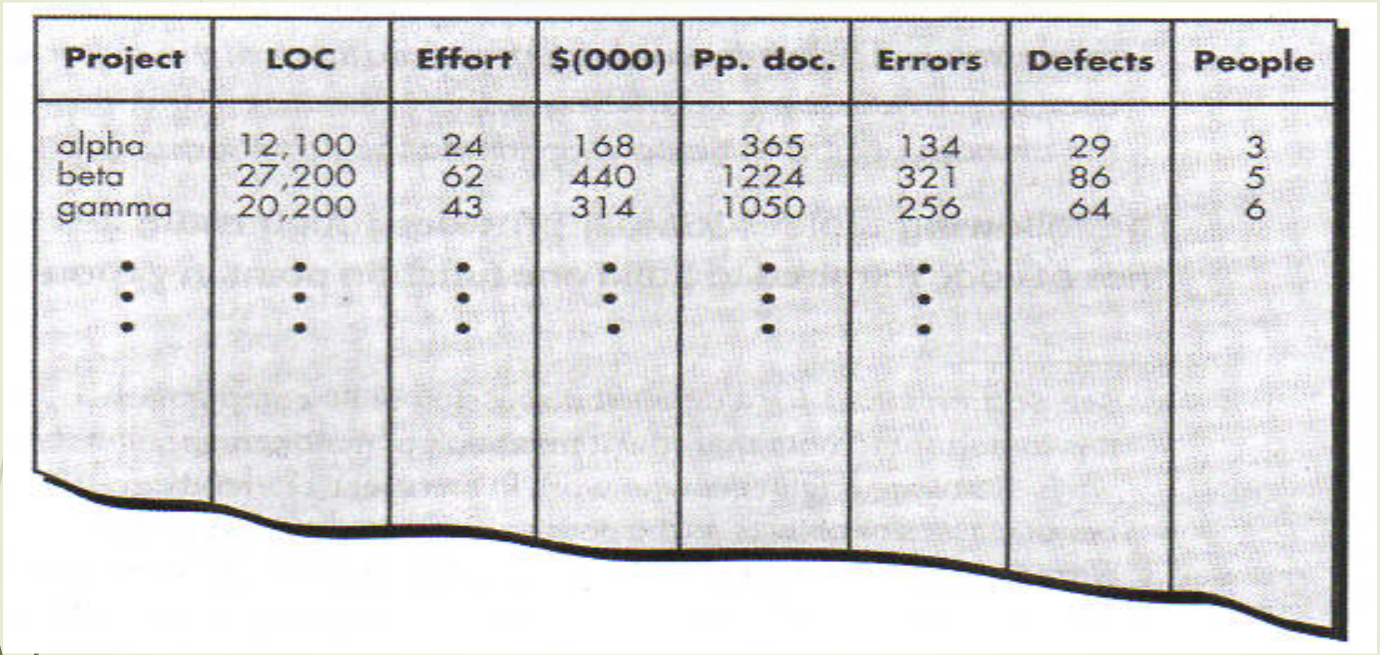
Derived by normalizing quality and/or productivity measures by considering the "*size*" of the software

Metrics include

Errors per KLOC	- Errors per person-month
Defects per KLOC	- KLOC per person-month
Dollars per KLOC	- Dollars per page of documentation
Pages of documentation per KLOC	

Opponents argue that KLOC measurements

- Are dependent on the programming language
- Penalize well-designed but short programs
- Cannot easily accommodate nonprocedural languages
- Require a level of detail that may be difficult to achieve



Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
⋮	⋮	⋮	⋮	⋮	⋮		

Function-Oriented Metrics

It is a measure of the functionality delivered by the application as a normalization value

Eg. Number of input, number of output, number of files, number of interfaces

Function-oriented Metrics

Function-oriented metrics use a measure of the functionality delivered by the application as a normalization value

Most widely used metric of this type is the function point:

FP = count total * [0.65 + 0.01 * sum (value adj. factors)]

Function point values on past projects can be used to compute, for example, the average number of lines of code per function point (e.g., 60)

Function Point Controversy

Like the KLOC measure, function point use also has proponents and opponents

Proponents claim that

- FP is programming language independent
- FP is based on data that are more likely to be known in the early stages of a project, making it more attractive as an estimation approach

Opponents claim that

- FP requires some “sleight of hand” because the computation is based on subjective data
- Counts of the information domain can be difficult to collect after the fact
- FP has no direct physical meaning…it’s just a number

Reconciling LOC and FP Metrics

- Relationship between LOC and FP depends upon
  - The programming language that is used to implement the software
  - The quality of the design
- FP and LOC have been found to be relatively accurate predictors of software development effort and cost
  - However, a historical baseline of information must first be established
- LOC and FP can be used to estimate object-oriented software projects
  - However, they do not provide enough granularity for the schedule and effort adjustments required in the iterations of an evolutionary or incremental process
- The table on the next slide provides a rough estimate of the average LOC to one FP in various programming languages

LOC Per Function Point

Language	Average	Median	Low	High
Ada	154	--	104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
Java	55	53	9	214
PL/1	78	67	22	263
Visual Basic	47	42	16	158

Object-oriented Metrics

Number of scenario scripts (i.e., use cases)

- This number is directly related to the size of an application and to the number of test cases required to test the system

Number of <u>key</u> classes (the highly independent components)

- Key classes are defined early in object-oriented analysis and are central to the problem domain
- This number indicates the amount of effort required to develop the software
- It also indicates the potential amount of reuse to be applied during development

Number of <u>support</u> classes

- Support classes are required to implement the system but are not immediately related to the problem domain (e.g., user interface, database, computation)
- This number indicates the amount of effort and potential reuse

Object-oriented Metrics

- Average number of support classes per key class
  - Key classes are identified early in a project (e.g., at requirements analysis)
  - Estimation of the number of support classes can be made from the number of key classes
  - GUI applications have between <u>two and three times</u> more support classes as key classes
  - Non-GUI applications have between <u>one and two times</u> more support classes as key classes
- Number of subsystems
  - A subsystem is an aggregation of classes that support a function that is visible to the end user of a system

Metrics for Software Quality

Correctness

- This is the number of defects per KLOC, where a defect is a verified lack of conformance to requirements
- Defects are those problems reported by a program user after the program is released for general use

Maintainability

- This describes the ease with which a program can be corrected if an error is found, adapted if the environment changes, or enhanced if the customer has changed requirements
- Mean time to change (MTTC) : the time to analyze, design, implement, test, and distribute a change to all users
  - Maintainable programs on average have a lower MTTC

Defect Removal Efficiency

Defect removal efficiency provides benefits at both the project and process level

It is a measure of the filtering ability of QA activities as they are applied throughout all process framework activities

- It indicates the percentage of software errors found before software release

It is defined as DRE = E / (E + D)

- E is the number of errors found before delivery of the software to the end user
- D is the number of defects found after delivery

As D increases, DRE decreases (i.e., becomes a smaller and smaller fraction)

The ideal value of DRE is 1, which means no defects are found after delivery

DRE encourages a software team to institute techniques for finding as many errors as possible before delivery