

Software Project Management

Ali Ameer Gondal

Assistant Professor

University of Engineering & Technology

Taxila, Pakistan

ali.ameer@uettaxila.edu.pk

27th Oct. 2011

Software Project Management

Lecture # 8 (Part-I)

Process and Project Metrics

Outline

- Metrics for Process and Projects
 - Introduction
 - Software Metrics
 - Process metrics
 - Project metrics
 - Direct and Indirect software measures
 - Software Measurement
 - Size-oriented Metrics
 - Function-oriented Metrics
 - Object-oriented Metrics
 - Use-case-oriented Metrics
 - Software Quality Metrics

Introduction

- Software process and project metrics are *quantitative measures* that enable software engineers to gain insight into efficacy of the software process and projects that are conducted using the process as a framework.

Introduction (Contd.)

- Basic quality & productivity data (called measures) are collected.
- These data are then analyzed, compared against past averages (for similar projects) , and assessed to determine whether productivity & quality improvements have occurred.

Introduction (Contd.)

- Metrics also indicate problem areas in processes and projects which can then be rectified.
- Project metrics contribute to development of process metrics.

Introduction - Basic Terms

- What is Measurement?
 - It is the act of obtaining a *measure*.
 - It is a mechanism of objective evaluation.
 - It enables managers and practitioners to improve software process.
 - It assists in planning, tracking and controlling a software project and assessing product quality.

Introduction - Basic Terms (Contd.)

- What is a Measure?
 - To ascertain or appraise by comparing to a standard [1]

OR

- A standard or unit of measurement; the extent, dimensions, capacity, etc., of anything, especially as determined by a standard [2]

[1]IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 729 1983.

[2]Engineering an Effective Measurement Program Course Notes, 1994.

Introduction - Basic Terms (Contd.)

- What is a Metric?
 - A metric is quantitative measure of the degree to which a system, component or process possesses a given attribute (IEEE definition)
 - A metric is the degree to which a particular subject possesses the quality that is being measured. It is based upon two or measures.
(Ref: <http://it.toolbox.com/blogs/dw-cents/measures-metrics-and-indicators-23543>)

Introduction - Basic Terms (Contd.)

- Difference between terms:
- Measure
 - It is a value which does not provide enough information to make meaningful decisions
- Metric
 - A comparison of two or more measures, e.g., temperature values over time, errors per KLOC
- Indicator
 - Compares a metric against a baseline or expected result and helps in making decisions
- Reference:
<http://www.stsc.hill.af.mil/crosstalk/1995/03/Measure.asp>

Reference:

<http://www.stsc.hill.af.mil/crosstalk/1995/03/Measure.asp>

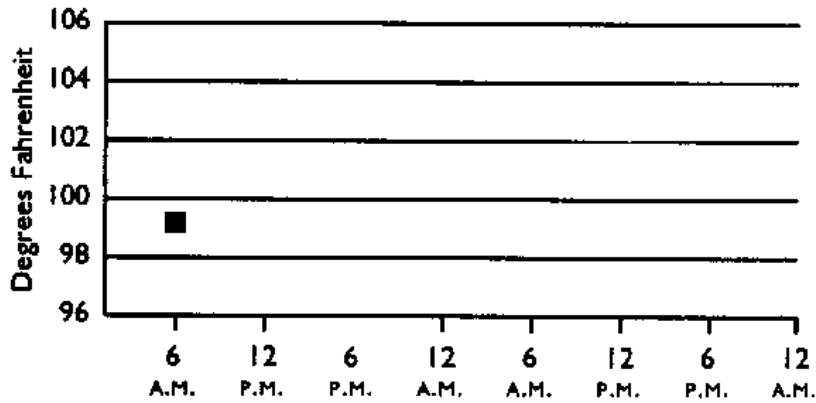


FIGURE 1: BODY TEMPERATURE (MEASURE).

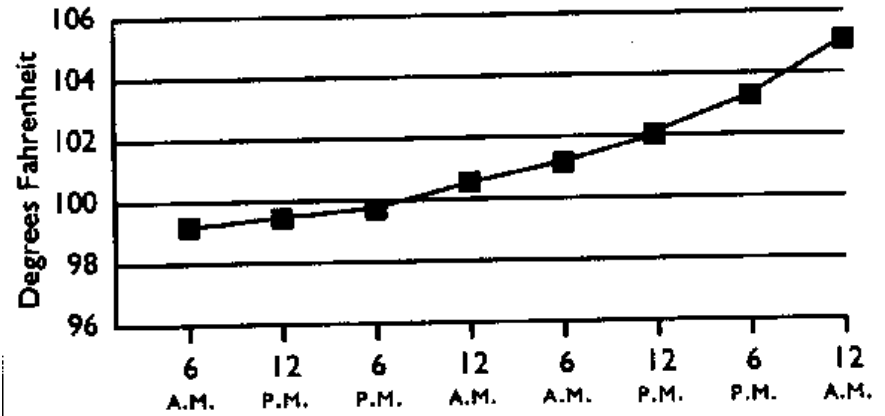


FIGURE 2: BODY TEMPERATURE (METRIC).

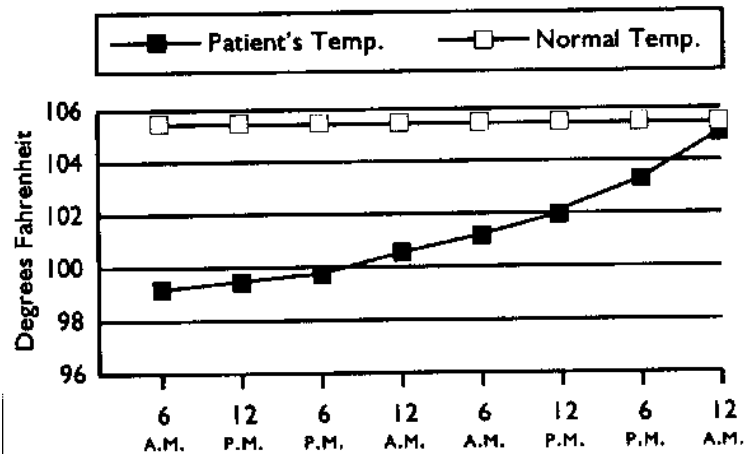


FIGURE 3: BODY TEMPERATURE COMPARED WITH NORMAL TEMPERATURE (INDICATOR).

Process Metrics & Software Process Improvement

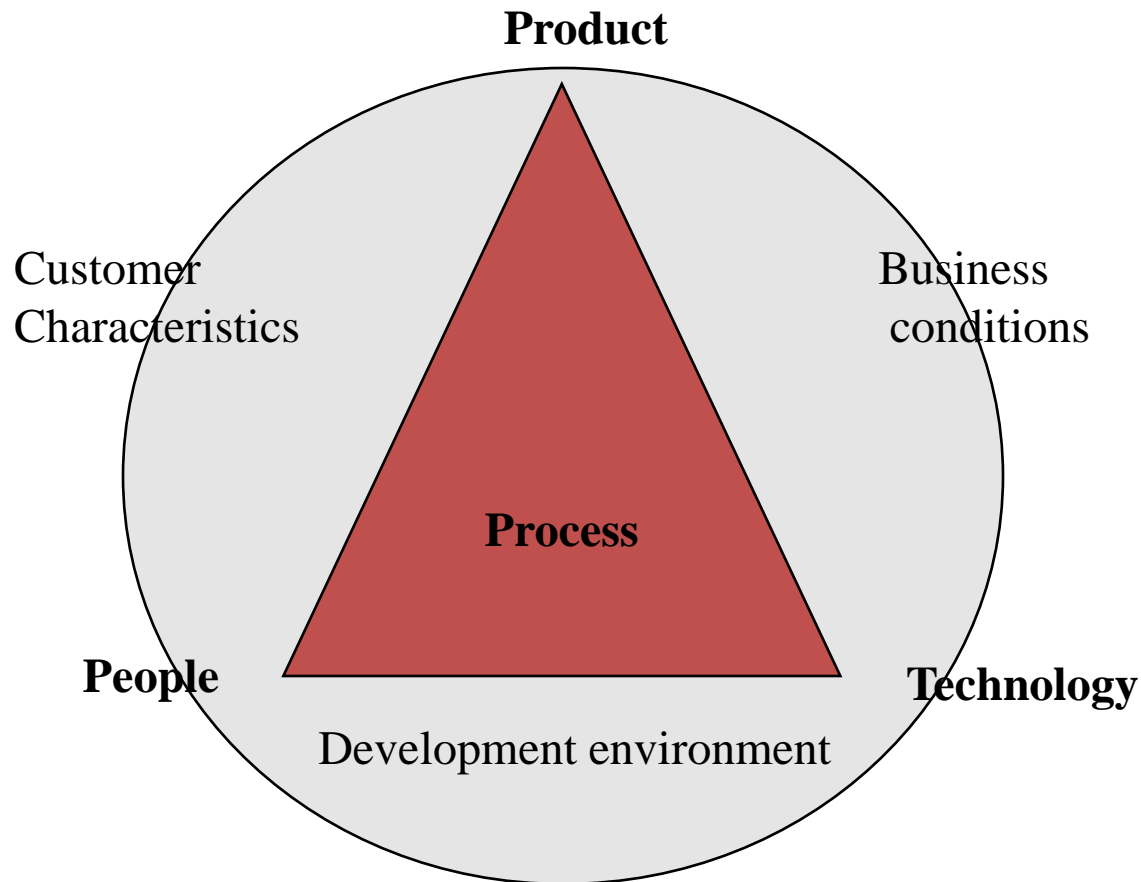
- To improve a process
 - Measure specific attributes of process
 - Develop meaningful metrics based on these attributes
 - These metrics provide indicators that lead to strategy for improvement
- Example
 - Metrics can be derived from the following process outcomes/attributes
 - Errors uncovered before software release,
 - defects delivered to & reported by end-user,
 - work products delivered,
 - Human efforts expended,
 - Calendar time expended,
 - Schedule conformance and other measures.

Improving Software Quality & Organizational Performance

- It is important to note that process is only one of the many controllable factors in improving software quality and organizational performance.
- The figure (on next slide) shows other factors connected with the ‘process’ that have significant influence on the same:
 - The skill and motivation of people
 - The complexity of the product
 - The technology

Improving Software Quality & Organizational Performance

Process connects 3 important factors that influence software quality and organizational performance



Project Metrics

- Used by project managers and software team to adapt project workflow and technical activities.
- Objective is
 - To minimize the development schedule by making adjustments that can avoid delays and mitigate potential problems.
 - To assess product quality on an ongoing basis and when necessary, modify technical approach to improve quality.
 - Most first application of project metrics on software projects occurs during estimation (of time, effort).
 - More project metrics are used as technical work of project commences.

Software Measurement

- Software measurement can be categorized in two ways
 - (1) direct measures of software process and product
 - (2) indirect measures of software product
- Direct Measures
 - of *software process* are
 - Cost and effort
 - of *software product* are
 - Line of Code (LOC)
 - Execution Speed
 - Defects over specified period of time

Software Measurement (Contd.)

- Indirect Measures
 - Of *product* are
 - Functionality
 - Quality
 - Complexity
 - Efficiency
 - Reliability
 - Maintainability
 - .. Other Abilities

Size Oriented Metrics

- These are derived by normalizing quality and/or productivity measures by considering the *size* of software that has been produced.
- Assume that an organization maintains record of the following size-oriented *measures* for each project completed:
 - Lines of code (LOC)
 - Effort in person-months
(see next slide)

Size Oriented Metrics (Contd.)

- Project cost (in dollars)
- Number of pages of documentation
- Number of errors
- Number of defects
- Number of people involved
- To develop metrics that can be incorporated with other similar projects, we choose LOC as a normalization factor.

Size Oriented Metrics (Contd.)

- Hence the following set of size-oriented metrics can be developed for each project:
 - Errors per KLOC (thousand lines of code)
 - Defects per KLOC
 - \$ per KLOC
 - Pages of documentation per KLOC
- Other metrics can be:
 - Errors per person-month
 - KLOC per person-month
 - \$ per page of documentation

Size Oriented Metrics - Pros and cons

- Size-oriented metrics are widely used but not universally accepted as best way to measure a software process
- Controversy lies in using LOC as a key measure
- Proponents claim
 - LOC is easy to count
 - Many existing estimation models use LOC or KLOC as key input
 - Large literature & data based on LOC exists

Size Oriented Metrics - Pros and cons

- But opponents argue that
 - LOC measures are programming language dependent
 - When considering productivity, LOC criteria penalizes well designed short programs
 - Can not accommodate non procedural languages
 - Planner must estimate LOC long before analysis and design

Function-oriented Metrics

- These use a measure of the functionality delivered by application as a normalization value.
- Most widely used function-oriented metric is function point (FP)
- FP's computation is based on characteristics of software information domain and complexity.

Function-oriented Metrics: Function Point

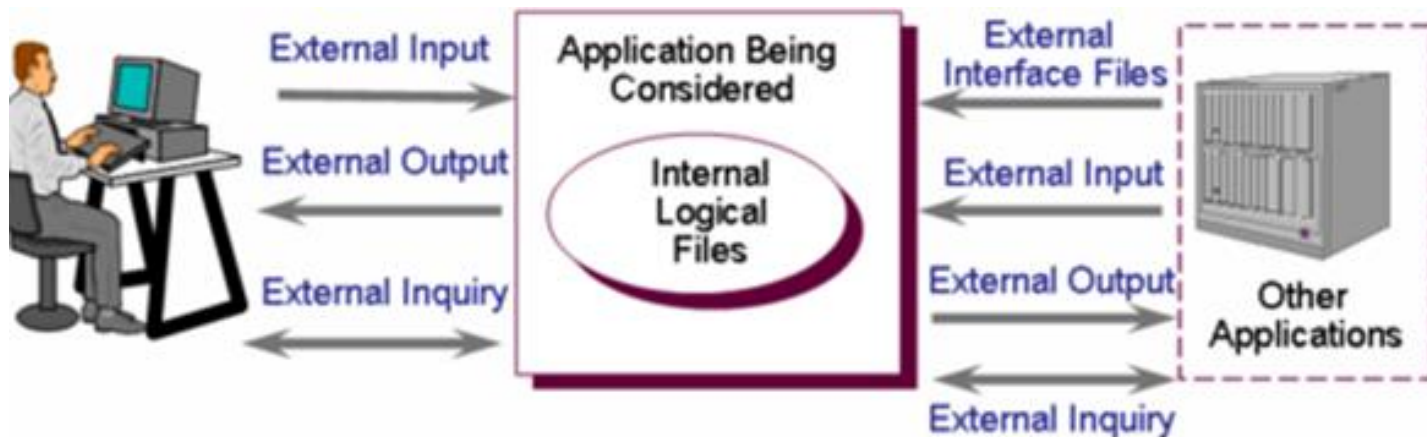
- **Function points are** a measure of the size of computer applications and the projects that build them.
- The size is measured from a functional or user, point of view.
- It is independent of the computer language, development methodology, technology or capability of the project team used to develop the application.

Function-oriented Metrics: Function Point (Contd.)

- First proposed by Albrecht; can be used to measure functionality delivered by a system.
- Using historical data, FP can be used to:
 - Estimate cost and effort required to design, code and test the software,
 - Predict no. of errors that will be encountered during testing,
 - Forecast the no. of components and/or project source lines in the implemented system.

Function Point Counting

- Function Type
 - Internal Logical File (ILF)
 - External Interface File (EIF)
 - External Inputs (EI)
 - External Outputs (EO)
 - External Inquiry (EQ)



Function Point Counting (Contd.)

Type of Component	Complexity of Components			Total
	Low	Average	High	
External Inputs	x 3 = _____	x 4 = _____	x 6 = _____	
External Outputs	x 4 = _____	x 5 = _____	x 7 = _____	
External Inquiries	x 3 = _____	x 4 = _____	x 6 = _____	
Internal Logical Files	x 7 = _____	x 10 = _____	x 15 = _____	
External Interface Files	x 5 = _____	x 7 = _____	x 10 = _____	
Total Number of Unadjusted Function Points				_____
Multiplied Value Adjustment Factor				_____
Total Adjusted Function Points				_____

Function Point Counting (Contd.)

General System Characteristic	
1.	Data communications
2.	Distributed data processing
3.	Performance
4.	Heavily used configuration
5.	Transaction rate
6.	On-Line data entry
7.	End-user efficiency
8.	On-Line update
9.	Complex processing
10.	Reusability
11.	Installation ease
12.	Operational ease
13.	Multiple sites
14.	Facilitate change

Function Point Counting (Contd.)

- Value Adjustment Factor (VAF)
 - Adjusts the Unadjusted FPC by +/- 35%
 - General System Characteristics each have a Degree of Influence (DI) factor of between 0 – 5
 - Add DI' s to get Total Degree of Influence (TDI)
 - $VAF = (TDI * 0.01) + 0.65$
- Adjusted Function Point Count
 - $AFPC = UFPC * VAF$

Function Point Counting (Contd.)

IFPUG Function Point Builder

Unadjusted Function Point Calculation:				Value Adjustment Factor Calculation:	
Function Type:	Functional Complexity:	Complexity Totals:	Function Type Total:	General System Characteristics:	Degree of Influence:
Internal Logical Files:	<input type="text" value="0"/> Low	$7 \times 0 = 0$	0	1. <u>D</u> ata Communications:	<input type="text" value="0"/>
	<input type="text" value="0"/> Avg	$10 \times 0 = 0$		2. Distributed Processing:	<input type="text" value="0"/>
	<input type="text" value="0"/> High	$15 \times 0 = 0$		3. <u>P</u> erformance:	<input type="text" value="0"/>
External Interface Files:	<input type="text" value="0"/> Low	$5 \times 0 = 0$	0	4. <u>H</u> eavily Used Configuration:	<input type="text" value="0"/>
	<input type="text" value="0"/> Avg	$7 \times 0 = 0$		5. <u>T</u> ransaction Rates:	<input type="text" value="0"/>
	<input type="text" value="0"/> High	$10 \times 0 = 0$		6. <u>O</u> n <u>L</u> ine Data Entry:	<input type="text" value="0"/>
External Inputs:	<input type="text" value="0"/> Low	$3 \times 0 = 0$	0	7. Design for End <u>U</u> ser Efficiency:	<input type="text" value="0"/>
	<input type="text" value="0"/> Avg	$4 \times 0 = 0$		8. <u>O</u> n-line Update:	<input type="text" value="0"/>
	<input type="text" value="0"/> High	$6 \times 0 = 0$		9. <u>C</u> omplex Processing:	<input type="text" value="0"/>
External Outputs:	<input type="text" value="0"/> Low	$4 \times 0 = 0$	0	10. Usable in Other <u>a</u> pplications:	<input type="text" value="0"/>
	<input type="text" value="0"/> Avg	$5 \times 0 = 0$		11. Installation Ease:	<input type="text" value="0"/>
	<input type="text" value="0"/> High	$7 \times 0 = 0$		12. Operational <u>E</u> ase:	<input type="text" value="0"/>
External Inquiries:	<input type="text" value="0"/> Low	$3 \times 0 = 0$	0	13. <u>M</u> ultiple Sites:	<input type="text" value="0"/>
	<input type="text" value="0"/> Avg	$4 \times 0 = 0$		14. <u>F</u> acilitate Change:	<input type="text" value="0"/>
	<input type="text" value="0"/> High	$6 \times 0 = 0$		Total Degree of Influence:	0
Unadjusted Function Point Count: 0					
Function Point Count:					

OK

Cancel

Function Point Counting (Contd.)

- **According to Pressman's Book:**
- Following empirical relationship is used to compute Function Point (FP)

$$FP = \text{count total} \times [0.65 + 0.01 \times \Sigma(F_i)]$$

- F_i ($i=1$ to 14) are VAF (Value adjustment factors) or simply 'adjustment values' based on the 14 characteristics in last slide
- These characteristics are on a scale of 0 (not applicable) to 5 (absolutely essential)

Function-oriented Metrics: Function Point (Contd.)

- Like LOC, FP is also controversial
 - Proponents claim
 - It is prog. Language independent, hence ideal for conventional and non-procedural languages
 - Based on data that can be known early phases of projects
 - Opponents claim
 - Computation is based on subjective rather than objective data
 - counts of information domain may be difficult to collect
 - FP is just a number and has no physical meaning

Object-oriented Metrics

- FP and LOC can be used to estimate OO software projects but these metrics do not address all aspects of such projects.
- Hence OO metrics are used for object-oriented projects
- Set of metrics for OO projects:
 - Number of scenario scripts
 - These are detailed sequence of steps about user and application interaction
 - These are directly correlated to application size and no. of test cases

Object-oriented Metrics (Contd.)

- Number of key classes
 - Key classes are independent components
 - They Indicate amount of development effort and potential reuse
- Number of support classes
 - Required to implement the system but not directly related to problem domain, e.g., UI classes, DB access, etc.
 - These indicate amount of development effort and potential reuse

Object-oriented Metrics (Contd.)

- Average number of support classes per key class
 - If these are known then estimation (based on total no. of classes) would be much simplified.
- Number of sub systems (aggregation of classes)
 - If identified, it is easier to lay out the schedule in which work on subsystems is partitioned among project staff.

Use-case oriented Metrics

- Use-cases describe user visible functions and features.
- They are defined early in software process and can be used as normalization measure before significant activities are initiated.
- They are independent of programming language
- No. of use cases directly proportional to
 - size of application in LOC &
 - no. of test cases that will be designed
- There is no standard size of a use case as they are created at different levels of abstraction
 - For this reason, it is a suspect as a normalization measure

Software Quality Metrics

- Measuring quality through
 - Correctness
 - It is degree to which software performs its required function
 - Common measure= defects per KLOC
 - For quality assessment defects are counted typically for 1 year
 - Maintainability
 - It is the ease with which a program can be
 - corrected if an error is found
 - Adapted if environment changes or
 - Enhanced if customer desires
 - Measured indirectly, e.g., Mean-time-to change (MTTC)
 - Maintainable programs -> lower MTTC

Software Quality Metrics

– Integrity

- System's ability to withstand (accidental & intentional) attacks
- Two more attributes that help determine integrity
 - threat = probability of attack within a given time (that causes failure)
 - security = probability that an attack will be repelled
- Integrity = $\Sigma [1 - (\text{threat} * (1 - \text{security}))]$

– Usability

- It quantifies ease of use and learning (time)

References

- Today's lecture taken from
 - Roger Pressman's "Software Engineering a practitioner's approach"
- For more in-depth study of function points, see <http://devdaily.com/fpa/fpa-borland/what-are-fps.shtml>
- SPM course material from University of Melbourne