

CHAPTER TWO

Software Metrics

- Measure : A measure provides a **quantitative indication** of the extent, amount, dimension, capacity , or size of some attribute of a product or process
- Metric : Metric as “ a **quantitative measure** of the **degree to** which a system, component, or process possesses a given attribute.
- Indicator: An indicator is a metric or condition of the metrics that provides **insight** into the software process, a software project, or the product itself

Software Metrics

- Metrics strongly support software project **management activities**
- They relate to the four functions of management as follows:
 - *Planning*
 - *Organizing*
 - *Controlling*
 - *Improving*

Size Estimation Metrics

- Size of a program is not the number **of bytes** that the source code occupies
- It is an indicator of the effort and time required to develop the program.
- Estimating the problem size is fundamental to estimating **the effort, time, and cost** of planned software.

Software Measurement

- Direct measure and indirect measure.
- Direct measures of the software include how many lines of **code (LOC) produced, execution speed, memory size, and defects reported.**
- Indirect measures include functionality, quality, complexity, efficiency, reliability, and maintainability of the software.

Lines of Code

- The simplest among all metrics available to estimate project size
- Project size estimated by counting the number of source instructions
- Lines used for commenting, header lines ignored
- To find LOC at the beginning of a project divide module into sub modules and so on until size of each module can be predicted

Disadvantages of LOC

- Gives a numerical value of problem size that vary widely with individual coding style

if(x>y)	x > y ? x++ : y++;
then x++;	
else	
y++;	

- Effort needed for analysis, design , coding, testing etc (not just coding)

Disadvantages of LOC

- # Larger Code size → Better Quality?
- # Logical Complexity?

Complex Logic → More Effort

Simple Logic → Less Effort

Disadvantage of LOC

- Accurate computation of LOC only after project completion!!

LOC Calculation

- **Cost per line of code** = Labor rate/
productivity
- **Estimated project cost** = cost per line of code
* estimated line of code
- **Estimated labor effort** = LOC/productivity

Example Of LOC

- Estimated line of code = 33,200
- Productivity = 620 LOC/pm
- Labor Rate per month = \$ 8000
- Cost per line of code = ?
- Estimated Project cost = ?
- Estimate Labor Effort = ?

Example of LOC

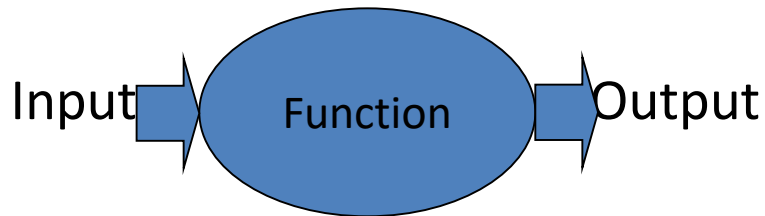
$$\square \text{Cost per line of code} = \$8000/620 \\ = \$13$$


$$\square \text{Estimated project cost} = 33,200 * 13 \\ = \$431,600$$

$$\square \text{Estimated effort} = \text{LOC} / \text{productivity} \\ = 33,200/620 \\ = 54 \text{ pm}$$

Function Point Metric

- Size of software product computed directly **from problem specification**
- Size of software = number of different functions/ features it supports



- Many features  Larger size
- Apart from that size depends on
 - number of files
 - number of interfaces
 - number of enquiries

Function Point

Size of Function Point (FP)= Weighted sum of these five problem characteristics

1.Number of inputs:

2.Number of Outputs:

3. Number of inquiries

4. Number of Files

5. Number of interfaces

Function Point

FIGURE 23.1

Computing
function points

Information Domain Value	Count	Weighting factor			
		Simple	Average	Complex	
External Inputs (EIs)	<input type="text"/> ×	3	4	6	= <input type="text"/>
External Outputs (EOs)	<input type="text"/> ×	4	5	7	= <input type="text"/>
External Inquiries (EQs)	<input type="text"/> ×	3	4	6	= <input type="text"/>
Internal Logical Files (ILFs)	<input type="text"/> ×	7	10	15	= <input type="text"/>
External Interface Files (EIFs)	<input type="text"/> ×	5	7	10	= <input type="text"/>
Count total	—————→				<input type="text"/>

Function point

Limitation of function point metric

- The weight of items on metric is fixed which may not sufficient for all cases.

Function Point

To compute function points (FP), the following relationship is used:

- **$FP = \text{count total} * [0.65 + 0.01 * \sum(F_i)]$**
- The F_i (i 1 to 14) are value adjustment factors (VAF) based on responses to the following questions

Function Point

- **These 14 questions are scaled to 0 to 5 where,**
- *0 = No influence or No importance, 1= Incidental, 2= moderate, 3= Average, 4 = Significant, 5 = Essential*

Function Point

1. Does the system require **reliable backup and recovery**?
2. Are specialized data communications required to transfer information to or from the application?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require online data entry?
7. Does the online data entry require the input transaction to be built over multiple screens or operations?
8. Are the ILFs updated online?

Function Point

- 9. Are the inputs, outputs, files, or inquiries complex?
- 10. Is the internal processing complex?
- 11. Is the code designed to be reusable?
- 12. Are conversion and installation included in the design?
- 13. Is the system designed for multiple installations in different organizations?
- 14. Is the application designed to facilitate change and ease of use by the user?

Function Point

- Example : With the given data for an online shopping site developed by ABC software developers
- Number of user input = 98
- Numbers of user Output = 51
- Number of User Inquires = 47
- Number of External Interfaces = 32
- Number of Logical Files = 61
- Assuming that the complexity of the given website development is average, compute the function point if the productivity of the ABC S/W developers is 35 FP/PM and their salary structure is Rs. 1500 per month on average, estimate total cost of the software .

Function Point

- Formula for Function Point
- **Cost per FP** = labor rate/productivity
- **Estimated project cost** = *estimated fP* cost per Fp*
- **Estimated Labor effort** = *Estimated FP/Productivity*

Function Point

- $FP = 672$
- Organizational productivity = 6.5 FP/pm
- Labor Rate per month = \$8000
- Cost per FP = $\$8000/6.5 = \1230
- Effort = $FP/Productivity = 672/6.5 = 103 \text{ PM}$
- Total project cost = $(672 * 1230) = \$826560$

FP Problem (Class Work)

- Assuming that the complexity of the NCIT MIS software development is average, compute the function point for it with the given data:
 - Number of User Input : 95
 - Number of User Output : 55
 - Number of Inquiries : 4
 - Number of logical Files : 66
 - Number of External Interfaces:27
- If the productivity of the software developers is 30 FP/PM and their salary structure is RS.18000 per month on average, estimate the total cost of the software.

Function point

Information domain value	Opt.	Likely	Pess.
Number of external inputs	20	24	30
Number of external outputs	12	15	22
Number of external inquiries	16	22	28
Number of internal logical files	4	4	5
Number of external interface files	2	2	3
<i>Count total</i>			

Fp Count

S*4

97

Function Point

- Expected Value for estimate variable size (s) = ?
- Make optimistic , most likely, pessimistic , estimate for each item, then compute expected value .

$$S = \frac{S_{opt} + 4S_m + S_{pess}}{6}$$

Categories of Metrics

- Product Metrics
- Process Metrics
- Project Metrics

Characteristics of Metrics

- Product Metrics : Product metric describes the characteristics of the product, such as **size, complexity, performance, efficiency.**

Characteristics of Metrics

Process Metrics: Process metric describe **effectiveness and quality** of the process. E.g

- **Effort** required in the process
- **Time** to produce the product
- Number of **defects** found during testing

Categories of Metrics

Project Metrics: Project metrics describe the project characteristics and execution.

E.g

- Number of software developer
- Staffing pattern over the life cycle of the software
- Cost and schedule
- Productivity

Attribute of Effective Software Metrics

1. Simple and computable
2. Empirically and Intuitively Persuasive (satisfy Engineers' intuitive)
3. Consistent and Objective
4. Consistent in the use of units and dimensions
5. Programming Language Independent
6. An Effective Mechanism for High Quality Feedback

Metric for Software Quality

- Software quality can be measured through the software Engineering process, **before release** to customer and **after release** to the customer
- The main goal of software engineering is to produce a high-quality system.
- A good software Engineer and good software engineering must measure if **high quality** is to be realized .

Metric for software quality

Measuring Quality

- Indicators to measure the quality
 - Correctness
 - Maintainability
 - Understandability
 - Integrity
 - Usability (user friendly or not)

Defect Removal Efficiency

- A quality metric that provides **benefit at both the project and process** level is defect removal efficiency (DRE)
- DRE is computed as

$$DRE = E/E+D$$

Where E = Errors found before delivery of the software

D = Defects found after the delivery

Metric for small Organization

organization might select the following set of easily collected measures:

- Time (hours or days) elapsed from the time a request is made until evaluation is complete, t_{queue} .
- Effort (person-hours) to perform the evaluation, W_{eval} .
- Time (hours or days) elapsed from completion of evaluation to assignment of

Metric for small Organization

- Change order to personnel, $teval$.
- Effort (person-hours) required to make the change, $Wchange$.
- Time required (hours or days) to make the change, $tchange$.
- Errors uncovered during work to make change, $Echange$.
- Defects uncovered after change is released to the customer base, $Dchange$.

Metric for small Organization

- Comprehensive software metrics programs
- Software organizations of all sizes measure
- Use the resultant metrics to help improve their local software.

COCOMO MODEL

- The constructive cost model (COCOMO) is an algorithmic software cost estimation model developed by **Barry Boehm**
- The model uses a basic regression formula, with some parameters that are derived from historical project data and current project characteristic

COCOMO II MODEL

- COCOMO II is actually a hierarchy of estimation models that address the following areas:
- **Application composition model:** Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
- **Early design stage model:** Used once requirements have been stabilized and basic software architecture has been established.
- **Post-architecture-stage model:** Used during the construction of the software.

COCOMO II Model

Types of COCOMO model levels

- *Basic COCOMO*
- *Intermediate COCOMO*
- *Detailed COCOMO*

Basic COCOMO-II Model

- COCOMO applies to three classes of software projects:
 - **Organic:** Developing well understood application programs, small experienced team
 - **Semi Detached:** mix of experienced and non-experienced team
 - **Embedded:** strongly coupled to computer hardware

Basic COCOMO

- $\text{Effort} = a (\text{KLOC})^b \text{ PM}$
- $\text{Time} = c (\text{Effort})^d \text{ Months}$
- $\text{Number of people required} = \frac{(\text{Effort applied})}{(\text{Development time})}$

COCOMO Model

Basic COCOMO

- $\text{Effort} = a (\text{KLOC})^b \text{ PM}$
- $\text{Time} = c (\text{Effort})^d \text{ Months}$
- $\text{Number of people required} = (\text{Effort applied}) / (\text{Development time})$

The program size is expressed in thousands of lines of code (KLOC). The coefficient a, b, c, and d are given as:

Software project	Effort schedule			
	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Example: The size of organic software is estimated to be 32,000 LOC. The average salary for software engineering is Rs. 15000/- per month. What will be effort and time for the completion of the project?

Solution:

- $\text{Effort applied} = 2.4 \times (32)^{1.05} \text{ PM} = 91.33 \text{ PM}$ (Since: 32000 LOC = 32KLOC)
- $\text{Time} = 2.5 \times (91.33)^{0.38} \text{ Month} = 13.899 \text{ Months}$
- $\text{Cost} = \text{Time} \times \text{Average salary per month} = 13.899 \times 15000 = \text{Rs. } 208480.85$
- $\text{People required} = (\text{Effort applied}) / (\text{development time}) = 6.57 = 7 \text{ persons}$

- **Define Measure and Metric with examples.** Given the data below, compute the function point value, productivity, documentation and cost per function for a project with the following information domain characteristics.
- Number of user inputs: 27
- Number of user outputs: 43
- Number of user inquiries: 5
- Number of files: 4
- Number of external interfaces: 2 and Effort = 37 PM,
Technical document=360 pages, user document = 129
pages, cost= Rs 200 per month complexity adjustment
values are 4,1,1,3,5,5,4,4,3,3,2,3,4,5