

(CT-21015)
Software Engineering
Mini Project Stage- II

Unit IV-Software Metrics

Unit IV-Software Metrics

Contents

- **Introduction to Software Metrics,**
- **Size-oriented metrics and function point metrics**
- **Effort and cost estimation techniques**
 - **LOC-based and Function-point based measures**
 - **The COCOMO model.**

Software Measurement

- Software Measurement is indicator who measures
 - Complete size of your project,
 - quantity amount – how many resources are required for particular project – (H/w or S/W or Human resources)
 - dimension of particular attribute of a product or process – Total time & cost required
- It helps the project manager & entire software team to take decisions that lead to successful completion of the project by generating quantity result.

Software measurements are of two categories:

- 1. Direct Measures:** It include software processes like Cost & Effort applied, Lines of code produced, Execution speed & Total no. of errors that have been reported.
- 2. Indirect Measures:** It include products like Functionality, Quality, Complexity, Reliability, Maintainability and many more.

Software Metric

- Software metric provide measures, functions or formulas for various aspects of software process & product.
- E.g. In software measurement we calculate total no. of errors occurred in particular project. But, In software metric we measure Total no. of errors occurred in per hour/per day in particular project.
- It including measuring software performance, planning work items, measuring productivity, and many other uses.

Software metrics are of three categories:

1. **Product Metrics:** It estimate Size, Complexity, Quality & Reliability of software.
2. **Process Metrics:** It estimate Faults rate during development, Pattern of testing defect arrival, Time required to fix a particular operation.
3. **Project Metrics:** It estimate Number of software developers, Cost, Scheduling and Productivity of software.

Size metrics : LOC

- Size oriented metrics concentrates on the size of program created.
- **LOC (Lines of Code) :**
 - Earliest and simple metric to calculate the size of computer program.
 - It is generally used in calculating and comparing the productivity of programmer.
 - **LOC does not count comment and blank lines in the code.**

Example:

Total Lines of Code (LOC) : 09

```
//Import header file
#include <iostream>
int main () {
    int num = 10;
//Logic of even number
    if (num % 2 == 0)
    {
        cout<<"It is even number";
    }
    return 0;
}
```

Size metrics : LOC

- **Advantages:**

- Most used metric in cost estimation
- It is very easy in estimating the efforts

- **Disadvantages:**

- It can not measure the size of specification.
- Bad software design may cause an excessive line of code.
- It is language dependent
- Users can not easily understand it.

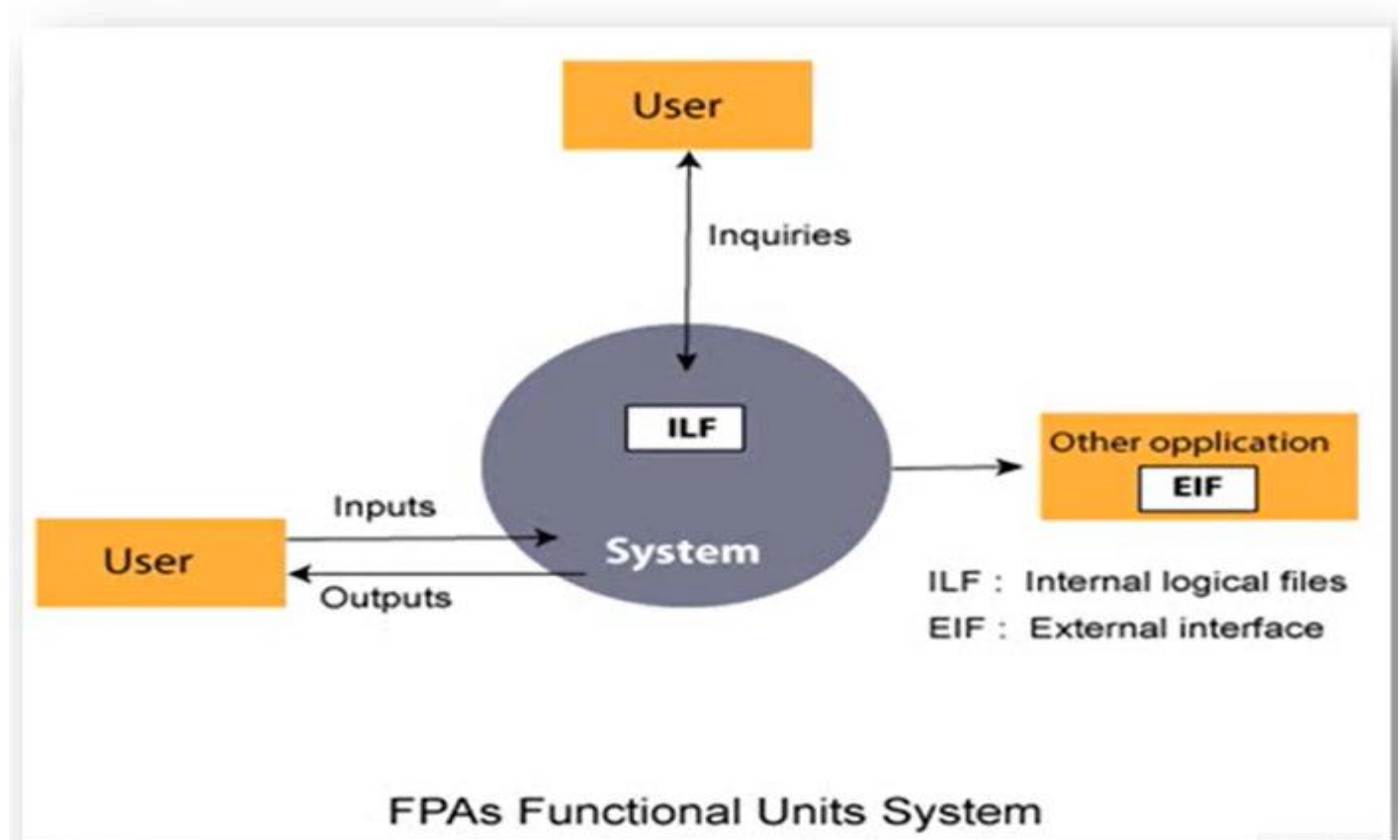
Project	LOC	Cost \$K	Efforts (Persons/ Month)	Documenta tion	Errors
A	10000	110	18	365	39
B	12000	115	20	370	45
C	15400	130	25	400	32

LOC Table

Size metrics : FP

- **FP (Functional Point) :**

- Function-oriented software metrics measure functionality, what the system performs, is the measure of the system size.
- It find out by counting the number and types of functions used in the applications



Size metrics : FP Cont...

FP Attributes

Measurements Parameters	Examples
1.Number of External Inputs(EI)	Input screen and tables
2. Number of External Output (EO)	Output screens and reports
3. Number of external inquiries (EQ)	Prompts and interrupts.
4. Number of internal files (ILF)	Databases and directories
5. Number of external interfaces (EIF)	Shared databases and shared routines.

Aspect	Function Points (FP)	Lines of Code (LOC)
Definition	Measures the functionality delivered to the user based on <u>user requirements</u> and <u>design</u> .	Measures the physical lines of source code written in a <u>programming language</u> .
Focus	User functionality and system behavior.	The amount of code written.
Language Independence	Independent of the programming language, platform, and technology used.	Dependent on the programming language, platform, and technology.
Measurement Criteria	Based on user inputs, outputs, inquiries, files, and interfaces.	Based on counting the lines of code written.
Calculation Method	Calculated using five main components: EI, EO, EQ, ILF, EIF, and adjusted using VAF.	Calculated by counting the number of lines in the codebase, excluding comments and blank lines.
Standardization	Internationally standardized by organizations like IFPUG (International Function Point Users Group).	No standardized method; varies widely depending on coding style and language.

Function Point Analysis (FPA)

Function Point Analysis (FPA) is a standardized method used to estimate the size, complexity, and functionality of software projects.

Unadjusted Function Points (UFP): Calculate UFP by multiplying the number of each type of component by its corresponding weight and summing them up.

Value Adjustment Factor (VAF): A factor that adjusts the UFP based on general system characteristics (GSCs), which include considerations like performance, usability, and complexity.

$$\text{FP} = \text{UFP} * (0.65 + 0.01 \times \text{Sum of GSCs})$$

5 Unadjusted Function Points

1) External Inputs (EI): 3

User registration form, product search form, add to cart form.

2) External Outputs (EO): 3

Confirmation email, order summary page, invoice generation.

3) External Inquiries (EQ): 2

Product availability check, order status check.

4) Internal Logical Files (ILF): 3

User data, product catalogue, order data.

5) External Interface Files (EIF): 2

Payment gateway interface, third-party shipping service interface

To get UFP - Multiply with weighing factor

Component	Low Complexity	Average Complexity	High Complexity
External Inputs (EI)	3 \times 3	4	6
External Outputs (EO)	4 \times 3	5	7
External Inquiries (EQ)	3 \times 2	4	6
Internal Logical Files (ILF)	7 \times 3	10	15
External Interface Files (EIF)	5 \times 2	7	10

Value Adjustment Factor (VAF)

General System Characteristics (GSCs):

- Data Communications
- Distributed Data Processing
- Performance
- Heavily Used Configuration
- Transaction Rate
- On-Line Data Entry
- End-User Efficiency
- On-Line Update
- Complex Processing
- Reusability
- Installation Ease
- Operational Ease
- Multiple Sites
- Facilitate Change

Value lies between 0 to 5

0 is low influence and 5 is highly influenced

$$0 \times 14 = 0$$
$$5 \times 14 = 70$$

So, Value of sum of GSC lies between 0 to 70

Consider a software project with the following information domain characteristic for the calculation of function point metric.

- 1 Number of external inputs (I) = $30 \times 4_+$
- 2 Number of external output (O) = $60 \times 5_+$
- 3 Number of external inquiries (E) = $23 \times 4_+$
- 4 Number of files (F) = $08 \times 10_+$
- 5 Number of external interfaces (N) = $02 \times 7 = 606$

It is given that the complexity weighting factors for I, O, E, F, and N are 4, 5, 4, 10, and 7, respectively. It is also given that, out of fourteen value adjustment factors that influence the development effort, four factors are not applicable, each of the other four factors has value 3, and each of the remaining factors has value 4. The computed value of the function point metric is _____.

$$FP = \underbrace{UFP}_{606} \times VAF \checkmark$$

$$VAF = \left[.65 + \frac{0.01 \times (\sum GSC)}{1.01} \right]$$

$$= \left[.65 + \frac{0.01 \times 36}{1.01} \right] = 1.01$$

$$FP = 606 \times 1.01 = 612$$

$$4 \times 0 + 4 \times 3 + 6 \times 4$$

$$0 + 12 + 24 = 36$$

If are Efforts = 36/Month then calculate the productivity

So,

$$P = 612/36 = 17$$

If cost is 1000 and productivity/month is 17 then

$$\text{Productivity-cost} = 17 \times 1000 = 17000$$

COCOMO Model

Effort & Cost Estimation Technique

COCOMO (Constructive Cost Model)

- Was first proposed by Dr. Barry Boehm in 1981.
- Is a heuristic estimation technique- this technique assumes that relationship among different parameters can be modeled using some mathematical expression.
- This approach implies that size is primary factor for cost, other factors have lesser effect.
- "constructive" implies that the complexity.
- COCOMO prescribes a 3-stage process for project estimation.
- An initial estimate is obtained, and over next two stages the initial estimate is refined to arrive at a more accurate estimate.

- Projects used in this model have following attributes :-
 1. ranging in size from 2,000 to 100,000 lines of code
 2. programming languages ranging from assembly to PL/I.
 3. These projects were based on the waterfall model of software development.
- Boehm stated that any software development project can be classified into three categories :-

Software Project Types

Mode	Project size	Nature of Project	Innovation	Deadline of the project	Development Environment
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & In house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer Interfaces required

Software Project Types

In COCOMO, projects are categorized into three types:

1. Organic Type:

- Project is small and simple. (2-50 KLOC)
- Few Requirements of projects.
- Project team is small with prior experience.
- The problem is well understood and has been solved in the past.
- **Examples:** Simple Inventory Management Systems & Data Processing Systems.

Software Project Types

2. Semidetached Type:

- Medium size and has mixed rigid requirements. (50-300 KLOC)
- Project has Complexity not too high & low.
- Both experienced and inexperienced members in Project Team.
- Project are few known and few unknown modules.
- **Examples:** Database Management System, Difficult inventory management system.

3. Embedded Type:

- Large project with fixed requirements of resources. (More than 300 KLOC)
- Larger team size with little previous experience.
- Highest level of complexity, creativity and experience requirement.
- **Examples:** ATM, Air Traffic control, Banking software.

Types of COCOMO Model



Type 1: Basic COCOMO Model

- It is type of static model to estimates software development effort quickly and roughly.
- It mainly deals with the number of lines of code in project.

Formula:

Effort (E) = $a \cdot (KLOC)^b$ MM

Scheduled Time (D) = $c \cdot (E)^d$ Months(M)

Where,

- **E** = Total effort required for the project in Man-Months (MM).
- **D** = Total time required for project development in Months (M).
- **KLOC** = The size of the code for the project in Kilo lines of code.
- **a, b, c, d** = The constant parameters for a software project.

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Type 1: Basic COCOMO Model Example

□ Problem Statement:

➤ Consider a software project using semi-detached mode with 300 Kloc.

Find out Effort estimation, Development time and Person estimation.

□ Solution:

Effort (E) = $a * (KLOC)^b = 3.0 * (300)^{1.12} = 1784.42 \text{ PM}$

Development Time (D) = $c(E)^d = 2.5(1784.42)^{0.35} = 34.35 \text{ Months(M)}$

Person Required (P) = $E/D = 1784.42/34.35 = 51.9481 \text{ Persons} \sim 52 \text{ Persons}$

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Type 2: Intermediate COCOMO Model

- Extension of Basic COCOMO model which enhance more accuracy to cost estimation model result.
- It include cost drivers (Product, Hardware, Resource & project Parameter) of project.

Formula:

Effort (E) = $a * (KLOC)^b * \underline{EAF}$ MM

Scheduled Time (D) = $c * (E)^d$ Months(M)


Where,

- **E** = Total effort required for the project in Man-Months (MM).
- **D** = Total time required for project development in Months (M).
- **KLOC** = The size of the code for the project in Kilo lines of code.
- **a, b, c, d** = The constant parameters for the software project.

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Type 2: Intermediate COCOMO Model

- **EAF: Effort Adjustment Factor**, which is calculated by multiplying the parameter values of different cost driver parameters. For ideal, the value is 1.

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL 	HIGH	VERY HIGH
Product Parameter					
Required Software	0.75	0.88	1	1.15	1.4
Size of Project Database	NA	0.94		1.08	1.16
Complexity of The Project	0.7	0.85		1.15	1.3

Hardware Parameter					
Performance Restriction	NA	NA	1	1.11	1.3
Memory Restriction	NA	NA		1.06	1.21
virtual Machine Environment	NA	0.87		1.15	1.3
Required Turnabout Time	NA	0.94		1.07	1.15

Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	NA

Project Parameter					
Software Engineering Methods	1.24	1.1	1	0.91	0.82
Use of Software Tools	1.24	1.1		0.91	0.83
Development Time	1.23	1.08		1.04	1.1

Type 2: Intermediate COCOMO Model Example

□ Problem Statement:

- For a given Semidetached project was estimated with a size of 300 KLOC. Calculate the Effort, Scheduled time for development by considering developer having high application experience and very low experience in programming.

□ Solution:

$$EAF = 0.82 * 1.14 = 0.9348$$

$$\text{Effort (E)} = a * (\text{KLOC})^b * EAF = 3.0 * (300)^{1.12} * 0.9348 = 1668.07 \text{ MM}$$

$$\text{Scheduled Time (D)} = c * (E)^d = 2.5 * (1668.07)^{0.35} = 33.55 \text{ Months(M)}$$

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	NA

Type 3: Detailed / Complete COCOMO Model

- The model incorporates all qualities of both Basic COCOMO and Intermediate COCOMO strategies on each software engineering process.
- The whole software is divided into different modules and then apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model

Type 3: Detailed COCOMO Model Example

□ Problem Statement:

A distributed Management Information System (MIS) product for an organization having offices at several places across the country can have the following sub-components:

- Database part
- Graphical User Interface (GUI) part
- Communication part

□ Solution:

- ✓ The communication part can be considered as Embedded software.
- ✓ The database part could be Semi-detached software,
- ✓ The GUI part Organic software.

The costs for these three components can be estimated separately and summed up to give the overall cost of the system.

Advantages of COCOMO Model

1. Provides a systematic way to estimate the cost and effort of a software project.
2. Estimate cost and effort of software project at different stages of the development process.
3. Helps in identifying the factors that have the greatest impact on the cost and effort of a software project.
4. Provide ideas about historical projects.
5. Easy to implement with various factors.

Disadvantages of COCOMO Model

1. It ignores requirements, customer skills and hardware issues.
2. It limits the accuracy of the software costs.
3. It is based on assumptions and averages.
4. It mostly depends on time factors.
5. Assumes that the size of the software is the main factor that determines the cost and effort of a software project, which may not always be the case.