Software Measurement and Metrics

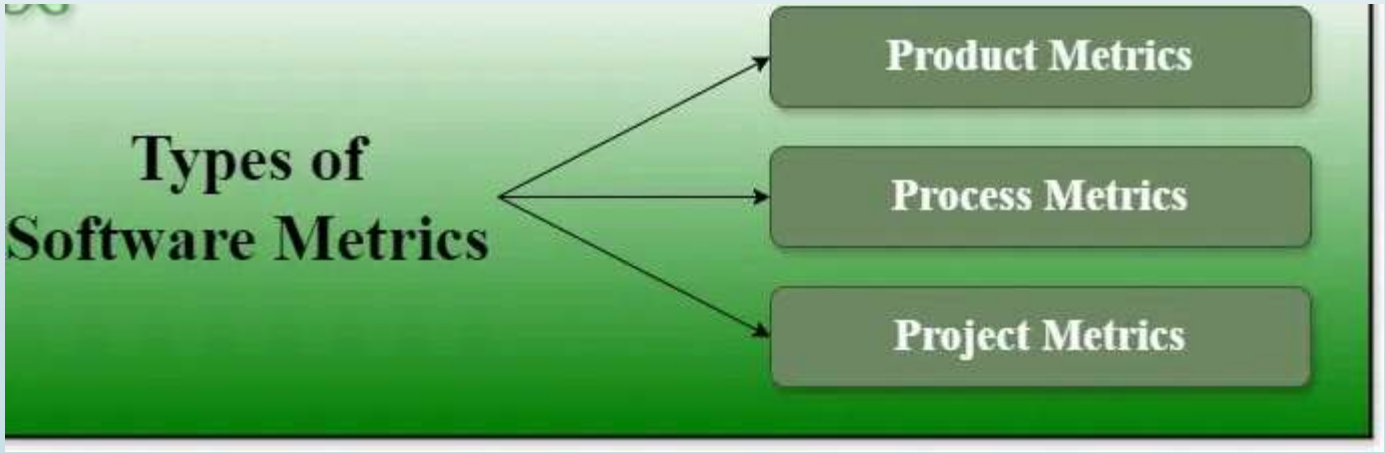
Ashima Tyagi
Assistant Professor
School of Computer Science & Engineering

Outline

- Software Measurement and Metrics
- Function Point (FP) Based Measures
- Practice questions
- Estimation of Various Parameters such as Cost, Efforts, Schedule/Duration
- Constructive Cost Models (COCOMO),
- Software Testing Tools

Software Metrics

- A software metric is a measure of software characteristics which are measurable or countable.
- Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, size, cost and many other uses.



Product Metrics

- Measure the **quality of the software product** (performance, reliability, maintainability).
- Used for **evaluating the software itself**.

Examples:

- Cyclomatic Complexity → Measures code complexity.
- Response Time → Measures system performance.
- Reliability Metrics → Failure rate, bug count, etc.
- Defect Density

Process Metrics

- Measure the **efficiency and effectiveness** of the software development process.
- Used for **process improvement and defect prevention**.

Examples:

- Defect Removal Efficiency (DRE) → Measures how well defects are removed before release.
- Mean Time to Failure (MTTF) → Average time before software fails.
- Review Efficiency → Measures the effectiveness of code reviews.

Project Metrics

- Measure **project management factors** like cost, schedule, and productivity.
- Used for **tracking project progress** and **resource allocation**.

Examples:

- Cost Variance (CV) → Difference between planned and actual cost.
- Schedule Variance (SV) → Difference between planned and actual schedule.
- Effort Estimation → Measures how much effort is needed to complete a project.

Size Metrics

- Size metrics measure the **size of software** in terms of

1. **lines of code (LOC)** or
2. **functional units**.

I. Lines of Code (LOC)

- Counts the **total lines of source code** in a program.
- Used for **effort estimation, cost prediction, and productivity analysis**.

Example:

Project	LOC	Developer Effort (Person-Months)
A	10,000	4
B	25,000	10

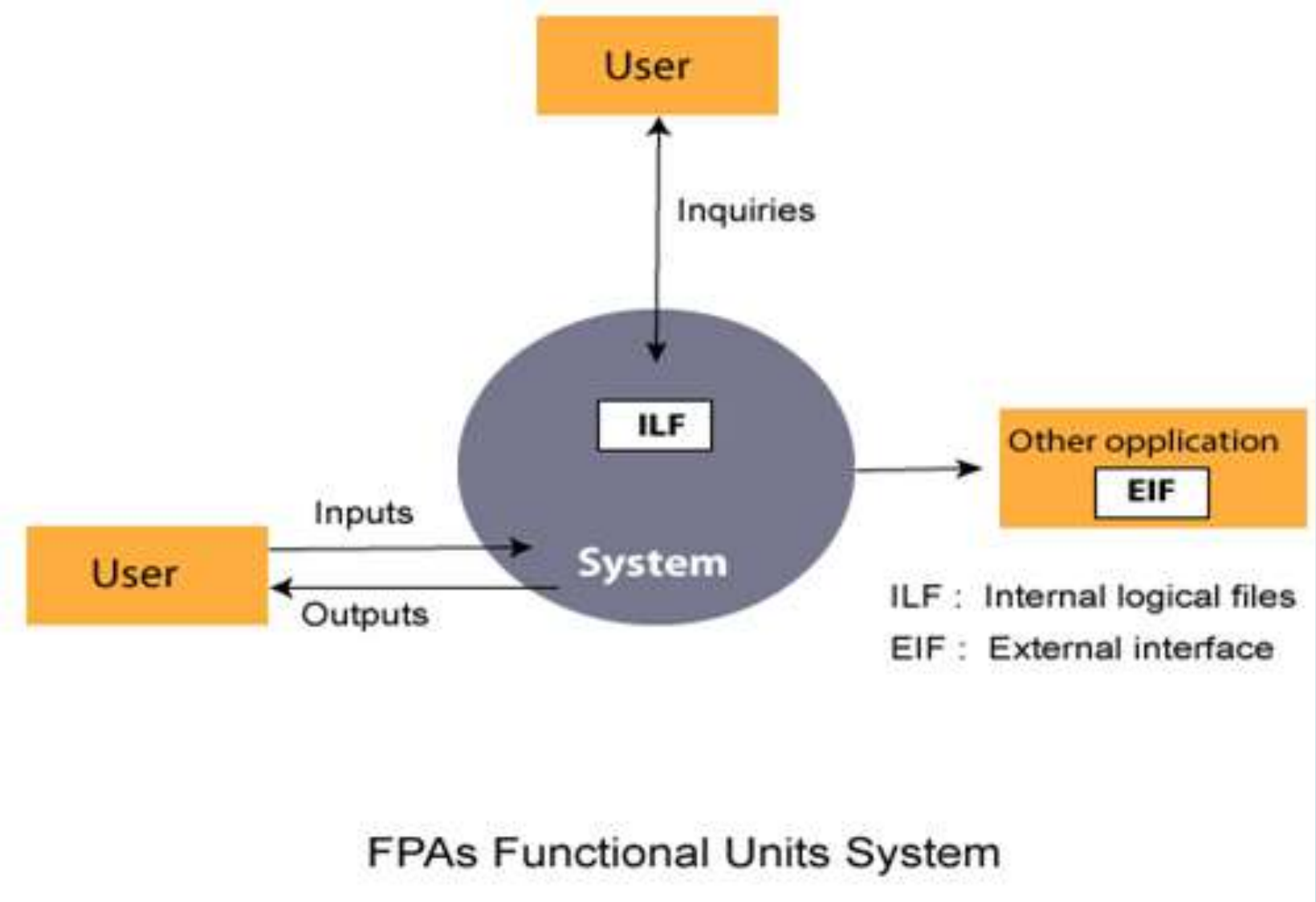
✕ Limitations of LOC:
•Does not measure **code quality**.
•Different programming languages have **different LOC counts** for the same functionality.

Functional Point (FP) Analysis

- The basic and primary purpose of the functional point analysis is to measure and provide the software application functional size to the client, customer, and the stakeholder on their request.
- Further, it is used to measure the software project development along with its maintenance, consistently throughout the project irrespective of the tools and the technologies.

Types of 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



FPAs Functional Units System

Weight of FP attributes

Measurement Parameter	Low	Average	High
1. Number of external inputs (EI)	3	4	6
2. Number of external outputs (EO)	4	5	7
3. Number of external inquiries (EQ)	3	4	6
4. Number of internal files (ILF)	7	10	15
5. Number of external interfaces (EIF)	5	7	10

Function Point(FP)= UFP(Unadjusted FP) x CAF

- F = 14 * scale

- Scale varies from 0 to 5 according to character of Complexity Adjustment Factor (CAF). Below table shows scale:

  0 - No Influence
  1 - Incidental
  2 - Moderate
  3 - Average
  4 - Significant
  5 - Essential

- Calculate Complexity Adjustment Factor (CAF).

  CAF = 0.65 + (0.01 * F)

Practice questions

1. **Given the following values, compute function point when all complexity adjustment factor (CAF) and weighting factors are average.**

- User Input = 50
- User Output = 40
- User Inquiries = 35
- User Files = 6
- External Interface = 4

Step-1: As complexity adjustment factor is average (given in question), hence, scale = 3.

- F = 14 * 3 = 42

Step-2:

- CAF = 0.65 + (0.01 * 42) = 1.07

Step-3: As weighting factors are also average (given in question) hence we will multiply each individual function point to corresponding values in TABLE.

- UFP = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628

Step-4:

- Function Point = 628 * 1.07 = 671.96

Function Units	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

2. Your company is developing a new software application, and you have been tasked with estimating the size of the project using the Function Point (FP) analysis. After a preliminary analysis, you have gathered the following information:

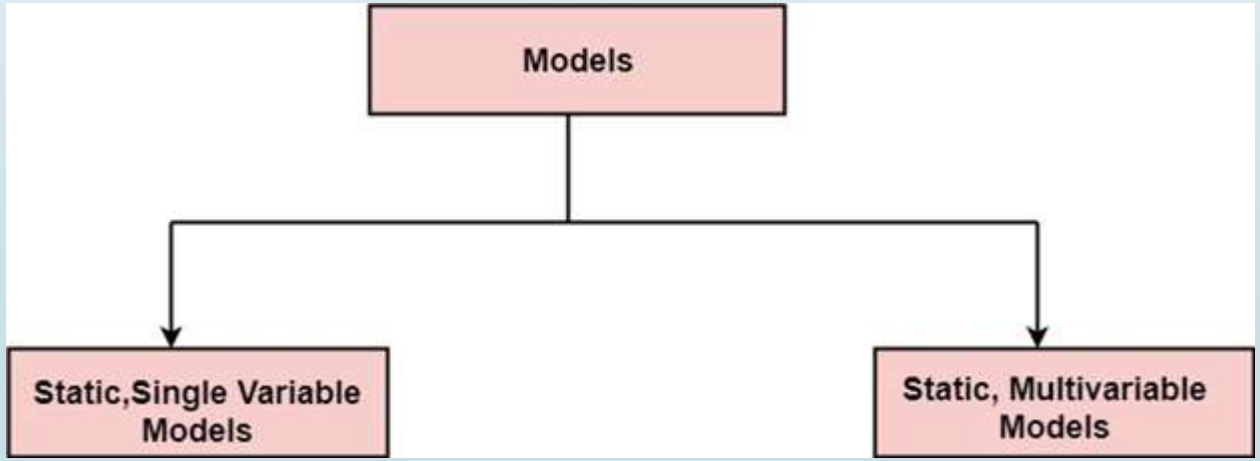
- Number of External Inputs (EI): 50, each of average complexity.
- Number of External Outputs (EO): 40, each of low complexity.
- Number of External Inquiries (EQ): 30, each of high complexity.
- Number of Internal Logical Files (ILF): 20, each of average complexity.
- Number of External Interface Files (EIF): 10, each of high complexity.
- Using the standard complexity weighting values:

- For EI, low=3, average=4, high=6.
- For EO, low=4, average=5, high=7.
- For EQ, low=3, average=4, high=6.
- For ILF, low=7, average=10, high=15.
- For EIF, low=5, average=7, high=10.

Evaluate the total function points (FP) for the project.

Cost Estimation Models

- A model may be static or dynamic.
- In a static model, a single variable is taken as a key element for calculating cost and time.
- In a dynamic model, all variable are interdependent, and there is no basic variable.



The Methods using this model utilize an equation to get the desired values such as cost, time, effort, etc. And these all depend on the same variable used as a predictor like, size.

This model is used to estimate the effort, cost and development time for a software project which depends on a single variable. The relationship is given by:

Cost (C) = a*(LOC)^b

Effort (E) = a*(LOC)^b PM

Development Time (DT) = a*(LOC)^b Months

Where LOC = Number of Lines of Code.

Below is an example of the most common equation:

$$C = aL^b$$

Where C is cost, L is size and a, b are constants.

We have an example of the static single-variable mode. e. i.e. **SEL model** which is used for estimating software production. The equation of this model is given below:

$$E = 1.4L^{0.93}$$

$$DOC = 30.4L^{0.90}$$

$$D = 4.6L^{0.26}$$

Where E is in Person-months, DOC i.e, documentation is in the number of pages and D is duration which is months.

Question 1:

A software project has an estimated size of 12,000 lines of code (LOC). The constants a=1.8 and b=1.15 are given. Estimate the cost of the project.

Solution:

We are given:

- a=1.8
- b=1.15
- L=12,000 LOC

Using the formula the cost of the project is **$38,826.94**.

Question 2:

A software project is estimated to have 8,000 lines of code (LOC). The cost constant a=2.2 and exponent b=1.3 are given. Calculate the cost of the project.

Solution:

We are given:

- a=2.2
- b=1.3
- L=8,000 LOC

Using the formula the estimated cost of the project is **$30,633.74**.

CCOMO Model

- **COCOMO (Constructive Cost Estimation Model)** is one of the most generally used software estimation models in the world.
- COCOMO predicts the efforts and schedule of a software product based on the size of the software.

Advantages :

- It works on historical data and provides more accurate details.
- Easy to implement with various factors. One can easily understand how it works.
- Easy to estimate the total cost of the project.
- The drivers are very helpful to understand the impact of the different factors that affect project crises.

Disadvantages :

- It ignores the hardware issues as well as the personal turnover level.
- It ignores all the documentation and requirements.
- It mostly depends on time factors.
- It limits the accuracy of software costs.
- It oversimplifies the impact of safety or security aspects.
- It also ignores customer skills, cooperation, and knowledge.

Basic COCOMO Model: The basic COCOMO model provides an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model:

Effort E = a(KLOC)^b PM

Time = c(Effort)^d Months

Person required = Effort/ time

Where

- KLOC is the estimated size of the software product indicate in Kilo Lines of Code,
- a and b are constants for each group of software products,
- time is the estimated time to develop the software, expressed in months,
- Effort is the total effort required to develop the software product, expressed in person months (PMs).

- **The above formula is used for the cost estimation of the basic COCOMO model and also is used in the subsequent models. The constant values a, b, c, and d for the Basic Model for the different categories of the system:**

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Practice questions

1. **Apply the COCOMO model to estimate the effort required for a project that is classified as 'organic' and consists of 32,000 lines of code (LOC). Use the basic COCOMO model formula. For an organic project, the typical values of A and B are 2.4 and 1.05**

Solution: Given Data

LOC = 32,000 A = 2.4 B = 1.05

Convert LOC to KLOC

$KLOC$=32,000/1,000=32

Calculate Effort

E=2.4×(32) ^1.05.

E≈ 91.33 Person-Months (PM)

Estimated Effort: The estimated effort required for the project is approximately 91.33 person-months.

2. A software development project is estimated to require 1,000 hours of development time. The hourly rate for developers is $50. Estimate the cost for the development of the software?

- Total Cost=Development Time×Hourly Rate
- Given:
- Development Time = 1,000 hours
- Hourly Rate = $50/hour
- **Calculation**

Total Cost=1,000 hours×50 dollars/hour

Total Cost=50,000 dollars

Estimated Cost

- The estimated cost for the development of the software is **$50,000**.

3. A software project is estimated to require 500 person-days of effort. If the team consists of 5 developers, solve how many days will it take to complete the project?

- Number of Days=Number of Developers/Total Person-Days
- Given:
- Total Person-Days = 500
- Number of Developers = 5
- **Calculation**

Number of Days=500 person-days / 5 developers

Number of Days=100 days

Result

- It will take 100 days to complete the project with a team of 5 developers.

4. **A software project has a fixed budget of $150,000. The hourly rate for developers is $75, and the estimated development time is 2,000 hours. Predict whether the project will stay within budget?**

Calculation

- **Total Cost of Development:** Total Cost=Hourly Rate × Estimated Development Time

Given:

- Hourly Rate = $75
- Estimated Development Time = 2,000 hours

Total Cost=75 dollars/hour×2,000 hours

Total Cost=150,000 dollars

- **Comparison**

- **Fixed Budget**: $150,000
- **Estimated Total Cost**: $150,000

Result

- The estimated total cost of development is equal to the fixed budget of $150,000. Therefore, the project is predicted to stay within budget if the estimated development time and hourly rate remain accurate.

5. Estimating Effort for a Semi-Detached Project A semi-detached software project has 50,000 LOC. Use:A=3.0 B=1.12. Find the Effort (in PM).

6. Estimating Effort and Time for an Embedded Project A project is classified as Embedded and has 100,000 LOC. Use:A=3.6 B=1.20. Also, use the Duration (Time in Months) formula:

Time=C×(Effort)^D

where for Embedded projects:

C=2.5

D=0.32

7. Finding LOC from Given Effort A project classified as Organic has an estimated Effort of 50 PM.Using:A=2.4 B=1.05 Find the estimated KLOC.

Software Testing Tools

Software testing tools help automate, manage, and execute tests efficiently.

Types of Testing Tools

Software testing is of two types, static testing, and dynamic testing.

1. Static Test Tools
2. Dynamic Test Tools

1. **Static Test Tools:** Static test tools are used to work on the static testing processes. In the testing through these tools, the typical approach is taken. These tools do not test the real execution of the software. Certain input and output are not required in these tools. Static test tools consist of the following:

- **Flow analyzers:** Flow analyzers provides flexibility in the data flow from input to output.
- **Path Tests:** It finds the not used code and code with inconsistency in the software.
- **Coverage Analyzers:** All rationale paths in the software are assured by the coverage analyzers.
- **Interface Analyzers:** They check out the consequences of passing variables and data in the modules.

2. **Dynamic Test Tools:** Dynamic testing process is performed by the dynamic test tools. These tools test the software with existing or current data. Dynamic test tools comprise the following:

- **Test driver:** The test driver provides the input data to a module-under-test (MUT).
- **Test Beds:** It displays source code along with the program under execution at the same time.
- **Emulators:** Emulators provide the response facilities which are used to imitate parts of the system not yet developed.
- **Mutation Analyzers:** They are used for testing the fault tolerance of the system by knowingly providing the errors in the code of the software.

Some popular testing tools:

1. **Jira**
2. **Selenium**
3. **Junit**

1. Jira (Test Management & Issue Tracking)

✅ What is Jira?

Jira is a test management and bug tracking tool widely used in Agile and DevOps environments.

✅ Key Features:

- Bug Tracking & Issue Management – Track software defects and improvements.
- Test Case Management (via plugins like Xray, Zephyr) – Manage test cases and execution.
- Agile Board Support – Scrum/Kanban support for sprint planning.
- Custom Workflows – Define test cycles, status, and issue resolution steps.

✅ Usage in Software Testing:

- Documenting test cases, linking them to user stories.
- Logging bugs and assigning them to developers.
- Managing test cycles and reporting defects.

2. Selenium (Automation Testing)

✅ What is Selenium?

Selenium is a popular automation testing tool for web applications.

✅ Components of Selenium:

- Selenium WebDriver – Automates web browser actions.
- Selenium Grid – Runs tests on multiple machines/browsers in parallel.
- Selenium IDE – A record-and-playback tool for simple tests.

✅ Use Cases:

- Automating UI testing for web applications.
- Performing cross-browser testing (Chrome, Firefox, Edge, etc.).
- Running data-driven tests using frameworks like TestNG or JUnit.

3. JUnit (Unit Testing in Java)

✅ What is JUnit?

JUnit is a **Java testing framework** used for unit testing. It ensures individual components of the software function correctly.

✅ Key Features:

- Annotations like @Test, @BeforeEach, @AfterEach.
- Assertions for expected vs. actual results.
- Parameterized Tests for multiple test cases.
- Test Suites to run multiple tests together.

Thank You