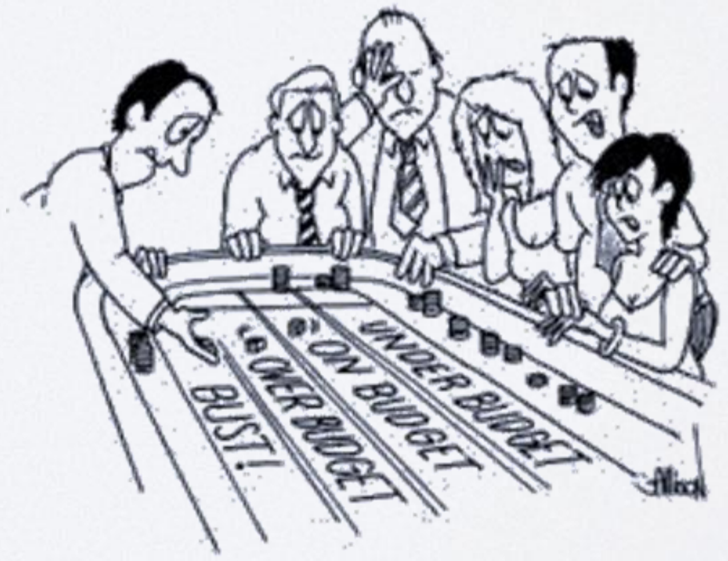
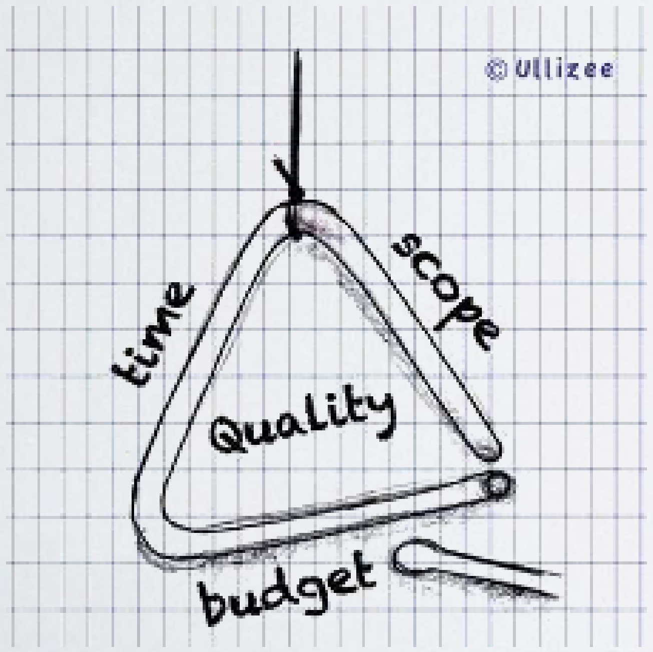
Estimation of Software Projects

Estimation (4 & 5)

- Estimation of resources, cost, and schedule for a software engineering effort requires
  - experience
  - access to good historical information (metrics)
  - the courage to commit to quantitative predictions when qualitative information is all that exists
- Estimation carries inherent risk and this risk leads to uncertainty

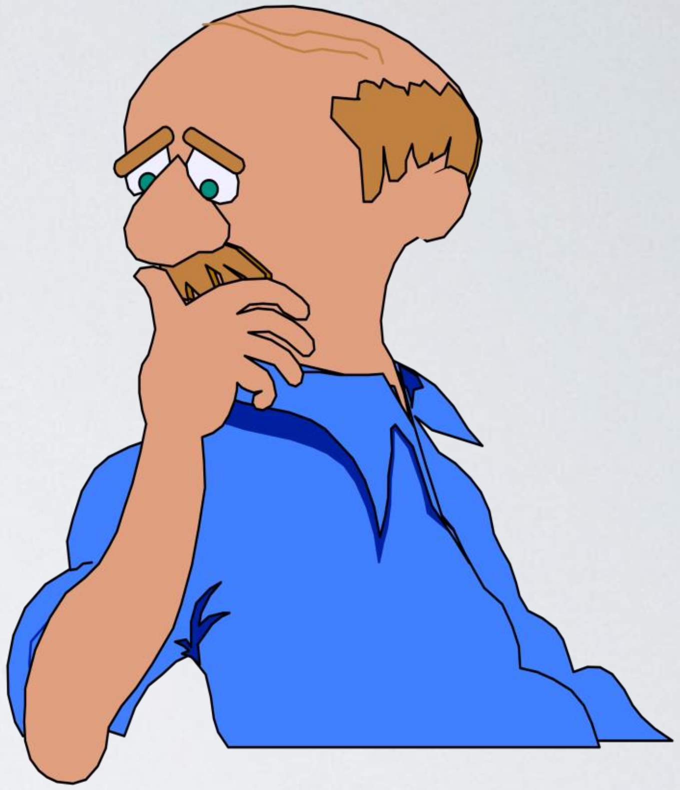


Project Estimation (5)

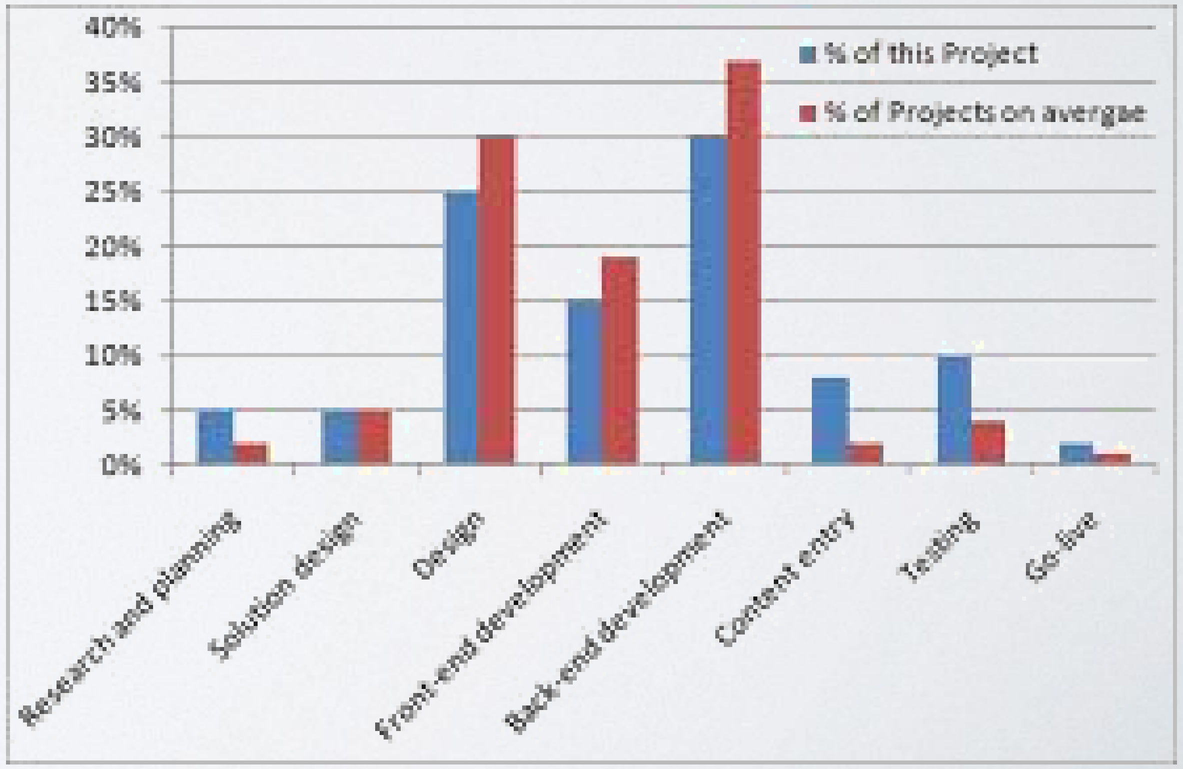


- ➢ Project scope must be understood
- ➢ Elaboration (decomposition) is necessary
- ➢ Historical metrics are very helpful
- ➢ At least two different techniques should be used
- ➢ Uncertainty is inherent in the process

Estimation Techniques (5)



- Past (similar) project experience
- Conventional estimation techniques
  - task breakdown and effort estimates
  - size (e.g., Function Point - FP) estimates
- Empirical models
- Automated tools



LINES OF CODE

- What is a LOC ?

  Declarations, Actual code including logic and computation

- What is NOT a LOC ?

1. BLANK Lines - included to improve readability of code
2. Comments - Included to help in code understanding as well as during maintenance.

Comments are NOT a LOC because it do not contribute to any kind of functionality, misused by developer a false notion about productivity.

EXAMPLE

```
for(int i = 1; i<10; i++)  //line of code 2

cout << i;



for(int i = 1; i<10;i++)

{

cout << i;

}

//line of code 4
```

ADVANTAGE

- Advantage of using LOC for size estimation
- Very easy to count and calculate from the developed code.

Disadvantage of using LOC for size estimation

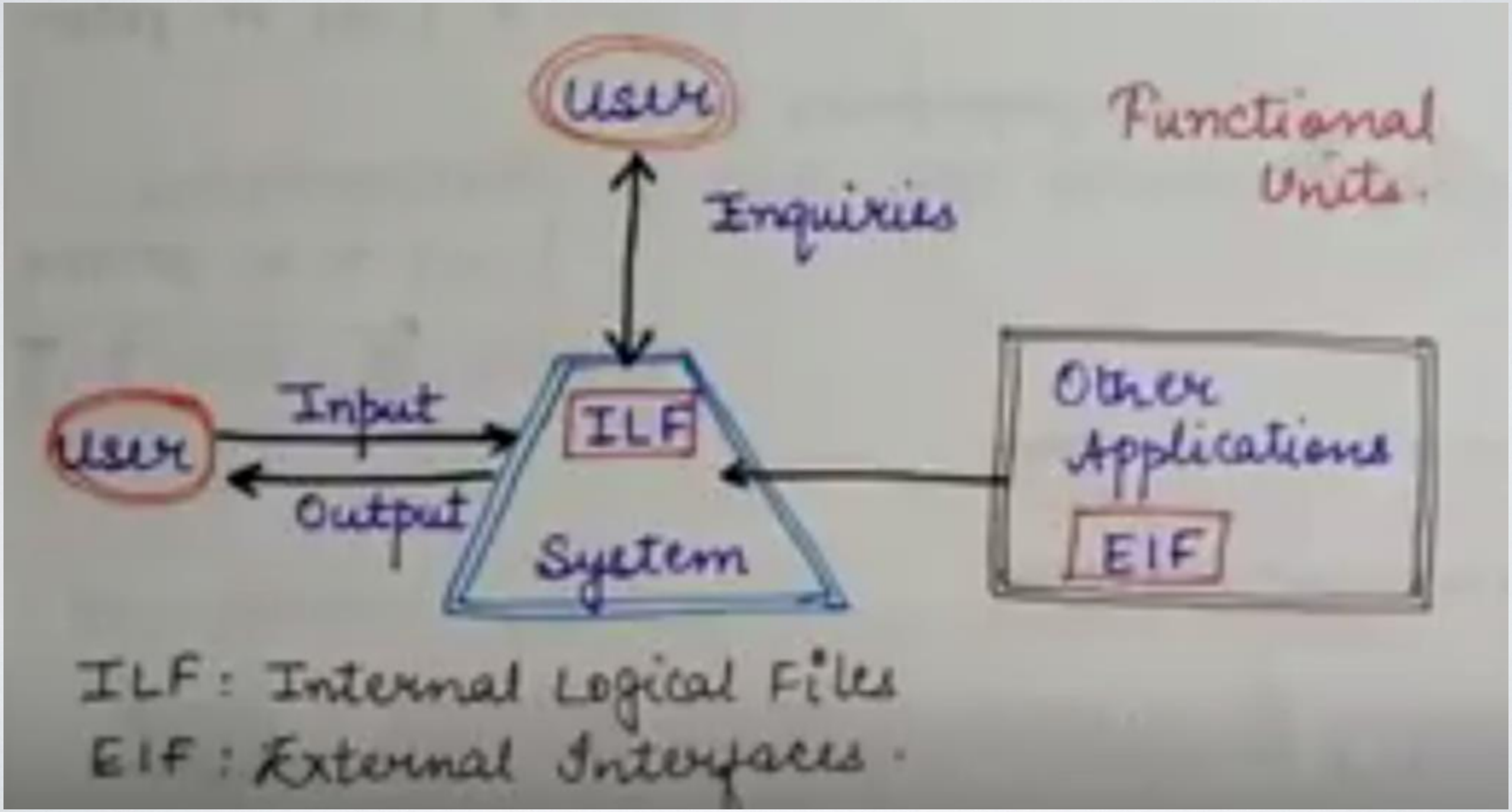
- It takes into account of coding effort only.
- What constitutes an LOC - Its varies form org to org.

FUNCTION BASED METRICS

FUNCTION POINTS

- It measures functionality from user’s point of view
- What the user receives from SW and What the user requests from SW.
- Focuses on what functionality is being delivered.

FUNCTIONAL UNITS



5 TYPES OF FUNCTIONAL UNITS

1. Internal Logic Files (ILF) - The control info or logically related data that is present within the system.

2. External Interface Files (EIF) - The control data or other logical data i.e referenced by the system but present in another system.

3. External Inputs (EI) - Data / control info that comes from outside our system

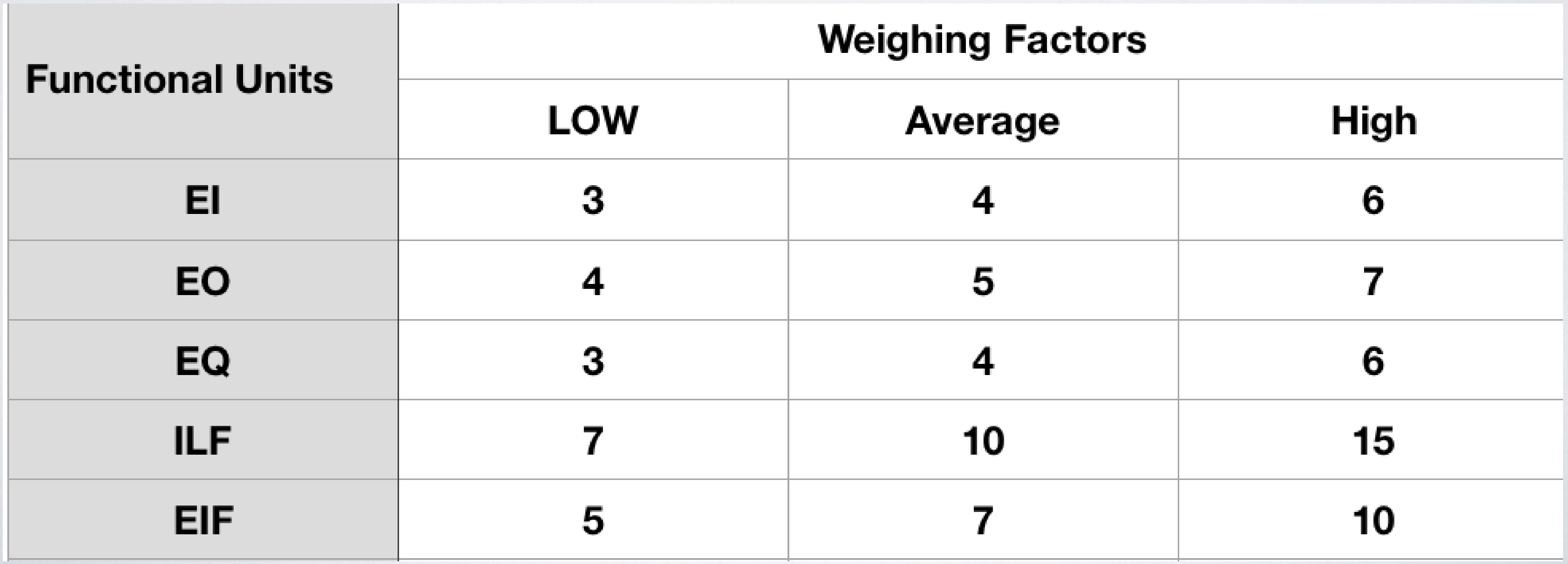
4. External Outputs (EO) - data that goes out of the system after generation

5. External Enquired (EQ) - Combination of i/o - o/p resulting data retrieval

FUNCTION POINTS

$$F.P = UFP \times CAF$$

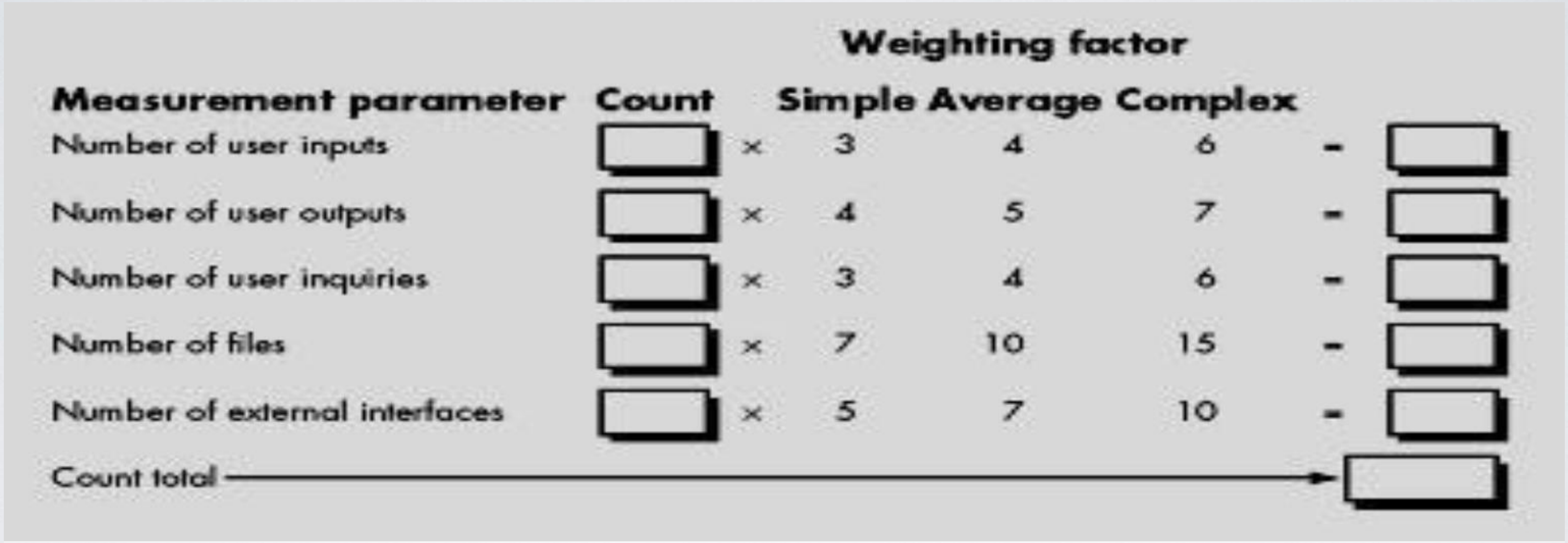
Step 1 : Each Function point is ranked according to complexity. There exists pre-defined weights



Functional Units	Weighing Factors		
	LOW	Average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

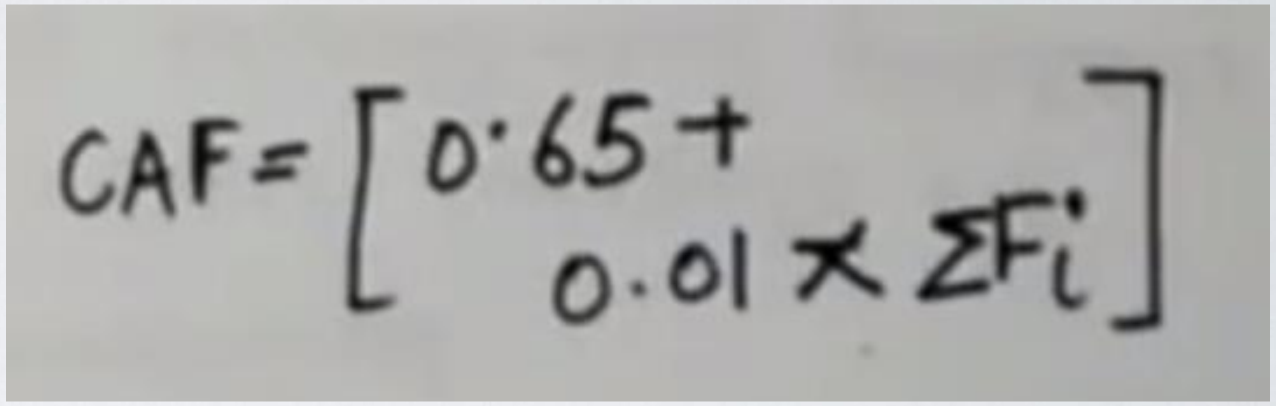
Step 2 : Calculate Unadjusted Function Point by multiplying each F.P by its corresponding weight factor

UFP = Sum of all the Complexities of all the EI's, EO's EQ's, ILF's and EIF's



				Weighting factor			
Measurement parameter	**Count**		**Simple**	**Average**	**Complex**		
Number of user inputs	☐	×	3	4	6	=	☐
Number of user outputs	☐	×	4	5	7	=	☐
Number of user inquiries	☐	×	3	4	6	=	☐
Number of files	☐	×	7	10	15	=	☐
Number of external interfaces	☐	×	5	7	10	=	☐
Count total	→	→	→	→	→	→	☐

Step 3 : Calculate Final Function Points



$$CAF = \left[0.65 + 0.01 \times \Sigma F_i \right]$$

Final F.P = UFP X CAF

- Complexity Adjustment Factor is calculated using 14 aspects of processing complexity.
- 14 questions answered on a scale of 0 - 5

0 - No Influences

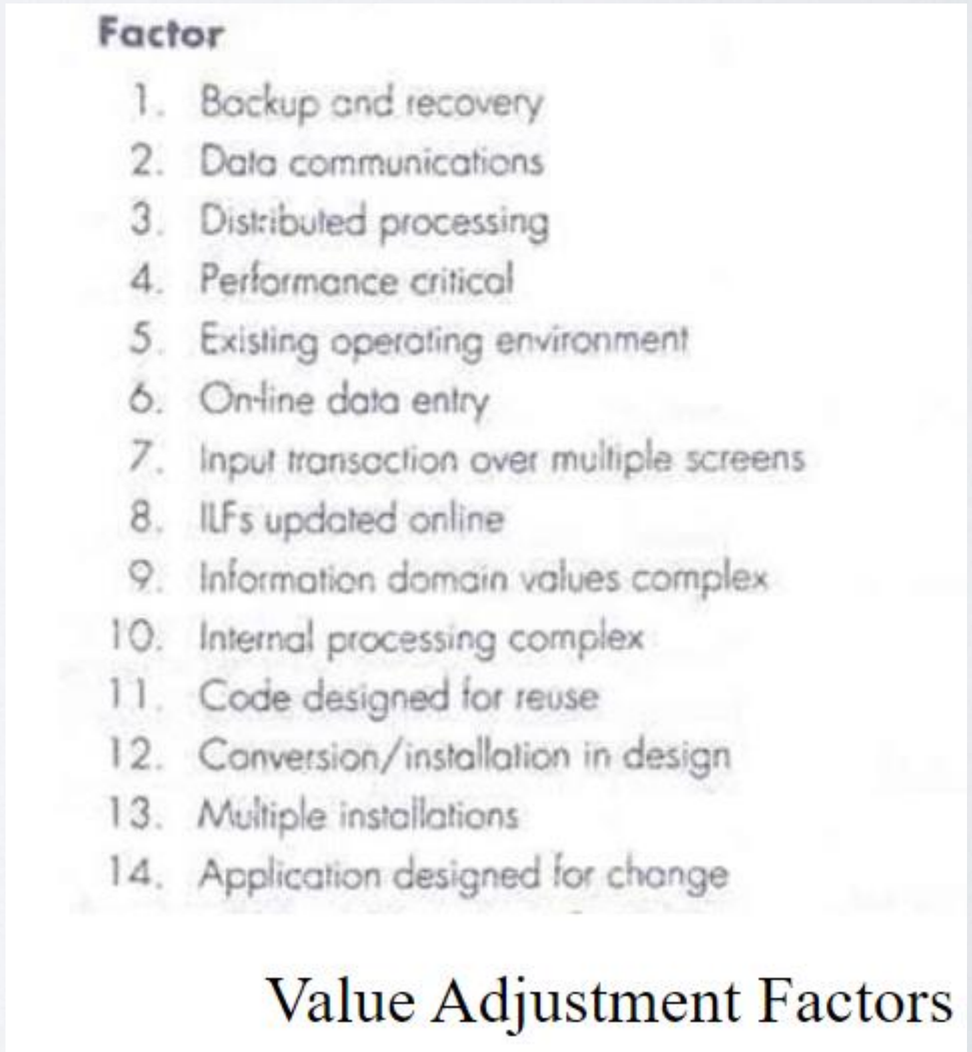
1 - Incidental

2 - Moderate

3 - Average

4 - Significant

5 - Essential



Factor

1. Backup and recovery
2. Data communications
3. Distributed processing
4. Performance critical
5. Existing operating environment
6. On-line data entry
7. Input transaction over multiple screens
8. ILFs updated online
9. Information domain values complex
10. Internal processing complex
11. Code designed for reuse
12. Conversion/installation in design
13. Multiple installations
14. Application designed for change

Value Adjustment Factors

EXAMPLE

Given the following values, compute F.P when all complexity adjustment factors and weighting factors are average.

User I/P = 50

User O/P = 40

User Enquires = 35

User Files = 6

External Interfaces = 4

SOLUTION

UFP = 50 x **4** + 40 x **5** + 35 x **4** + 6 x **10** + 4 x **7**

= 200 + 200 + 140 + 60 + 28 = 628

CAF = 0.65 + (0.01 x $\Sigma(F_i)$) = 0.65 + 0.01 (14 x 3) = 1.07

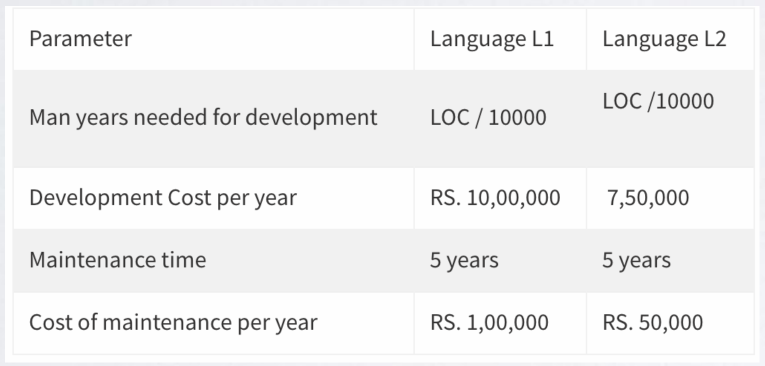
F.P = UFP x CAF = 628 x 1.07

ADVANTAGES OF F.P

- Size of sw delivered is measured independent of Language, technology and tools.
- F.P are directly estimated from requirements, before design and coding
  - We get an estimate of s/w size even before major design or coding happens (early phases)
  - Any change in requirements can be easily reflected in FP count

PROBLEM

Company needs to develop a strategy for software product development for which it has a choice of two programming languages L1 and L2. The number of lines of code (LOC) developed using L2 is estimated to be twice the LOC developed with L1. the product will have to be maintained for five years. Various parameters for the company are given in the table below.



Parameter	Language L1	Language L2
Man years needed for development	LOC / 10000	LOC /10000
Development Cost per year	RS. 10,00,000	7,50,000
Maintenance time	5 years	5 years
Cost of maintenance per year	RS. 1,00,000	RS. 50,000

Total cost of the project includes cost of development and maintenance. What is the LOC for L1 for which the cost of the project using L1 is equal to the cost of the project using L2?

(A) 4000 (B) 5000 (C) 4333 (D) 4667

ANSWER: (B)

EXPLANATION:

Let, LOC for L1=X

LOC for L2=2X

⇒ Development cost of L1 + Maintenance cost of

L1 = Development cost for L2 + Maintenance cost of L2

⇒ { [X / 10000] * 1000000 + 5 * 100000 } = { [2X / 10000] *

750000 + 5 * 50000}

⇒ 5X = 250000

⇒ X = 5000

Empirical Estimation Models(5)

General form:

$$\text{effort} = \text{tuning coefficient} * \text{size}^{\text{exponent}}$$

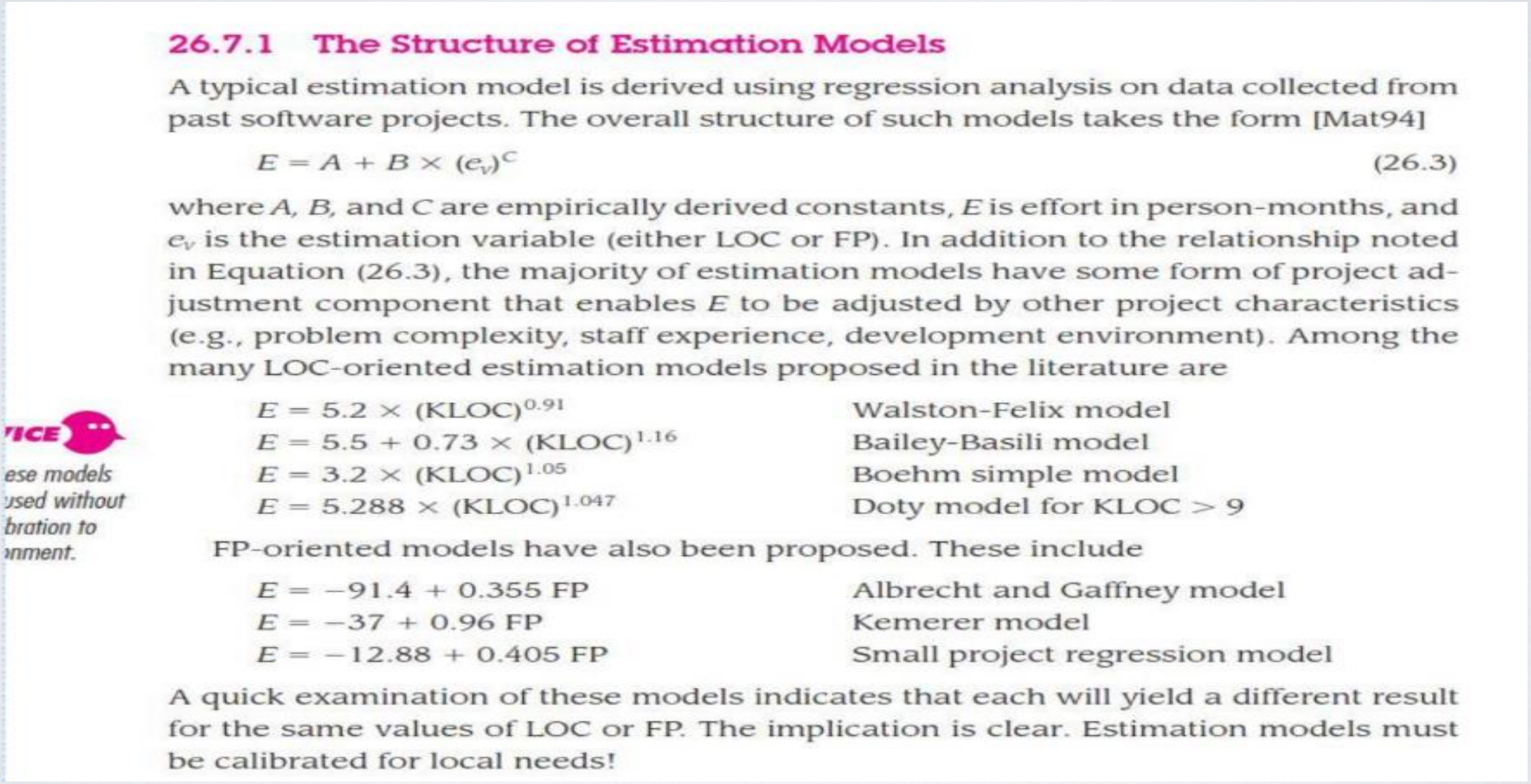
usually derived as person-months of effort required

either a constant or a number derived based on complexity of project

usually LOC but may also be function point

empirically derived

Empirical Estimation Models(5)



26.7.1 The Structure of Estimation Models

A typical estimation model is derived using regression analysis on data collected from past software projects. The overall structure of such models takes the form [Mat94]

$$E = A + B \times (e_v)^C \qquad (26.3)$$

where *A*, *B*, and *C* are empirically derived constants, *E* is effort in person-months, and e_v is the estimation variable (either LOC or FP). In addition to the relationship noted in Equation (26.3), the majority of estimation models have some form of project adjustment component that enables *E* to be adjusted by other project characteristics (e.g., problem complexity, staff experience, development environment). Among the many LOC-oriented estimation models proposed in the literature are

$E = 5.2 \times (\text{KLOC})^{0.91}$	Walston-Felix model
$E = 5.5 + 0.73 \times (\text{KLOC})^{1.16}$	Bailey-Basili model
$E = 3.2 \times (\text{KLOC})^{1.05}$	Boehm simple model
$E = 5.288 \times (\text{KLOC})^{1.047}$	Doty model for KLOC > 9

ICE

ese models used without bration to nment.

FP-oriented models have also been proposed. These include

$E = -91.4 + 0.355\ \text{FP}$	Albrecht and Gaffney model
$E = -37 + 0.96\ \text{FP}$	Kemerer model
$E = -12.88 + 0.405\ \text{FP}$	Small project regression model

A quick examination of these models indicates that each will yield a different result for the same values of LOC or FP. The implication is clear. Estimation models must be calibrated for local needs!

COCOMO

Constructive Cost Model

INTRODUCTION

Developed by: **Barry Boehm**

It is a **hierarchy** of software cost estimation models

1. Basic Model
2. Intermediate Model
3. Detailed Model

BASIC MODEL

- It estimates the software in a rough and quick manner.
- Mostly useful for small - medium sized software
- 3 modes of development.

a. Organic projects (Simple) - "small" teams with "good" experience working with "less than rigid" requirements

b. Semi-detached projects (Moderate- Medium) - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements

c. Embedded projects (Difficult/Complex) - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.(hardware, software, operational, ...)

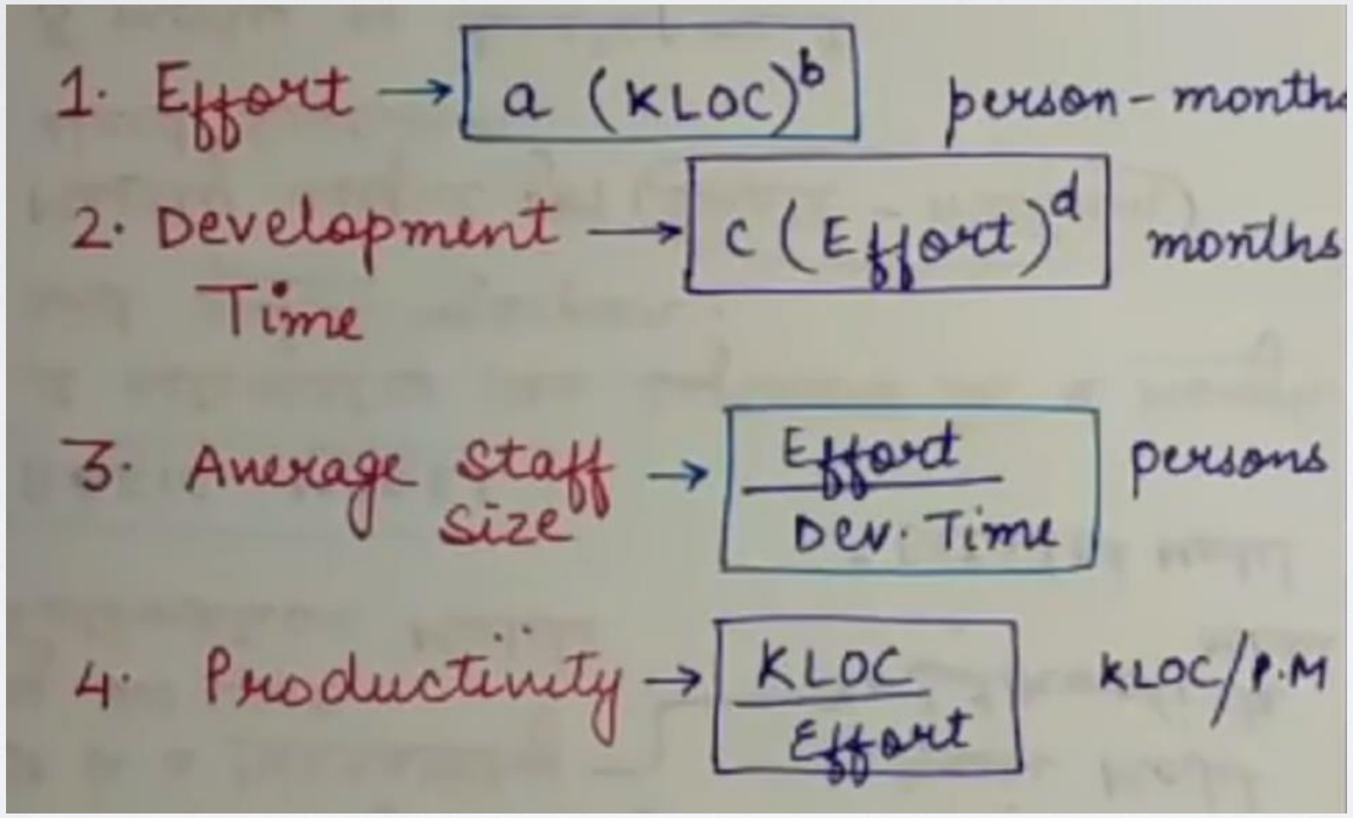
THE FEATURES OF COCOMO MODEL

- An empirical model based on project experience.
- Well-documented, ‘independent’ model which is not tied to a specific software vendor.
- Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2.
- COCOMO 2 takes into account different approaches to software development, reuse, etc.

WHEN ??

	ORGANIC	SEMI DETACHED	EMBEDDED
Size	2 - 50 KLOC	50 - 300 KLOC	300 & Above KLOC
Team Size	Small size	Medium size	Large size
Developer Experience	Experienced Developers needed	Average Experienced People	Very Little previous experience
Environment	Familiar environment	Less Familiar	Significant env. changes (almost new enviornment)
Innovation	Little	Medium	Major
Deadline	Not Tight	Medium	Tight
Eg.	Payroll System	Utility systems	Air Traffic Monitoring

BASIC MODEL EQUATIONS



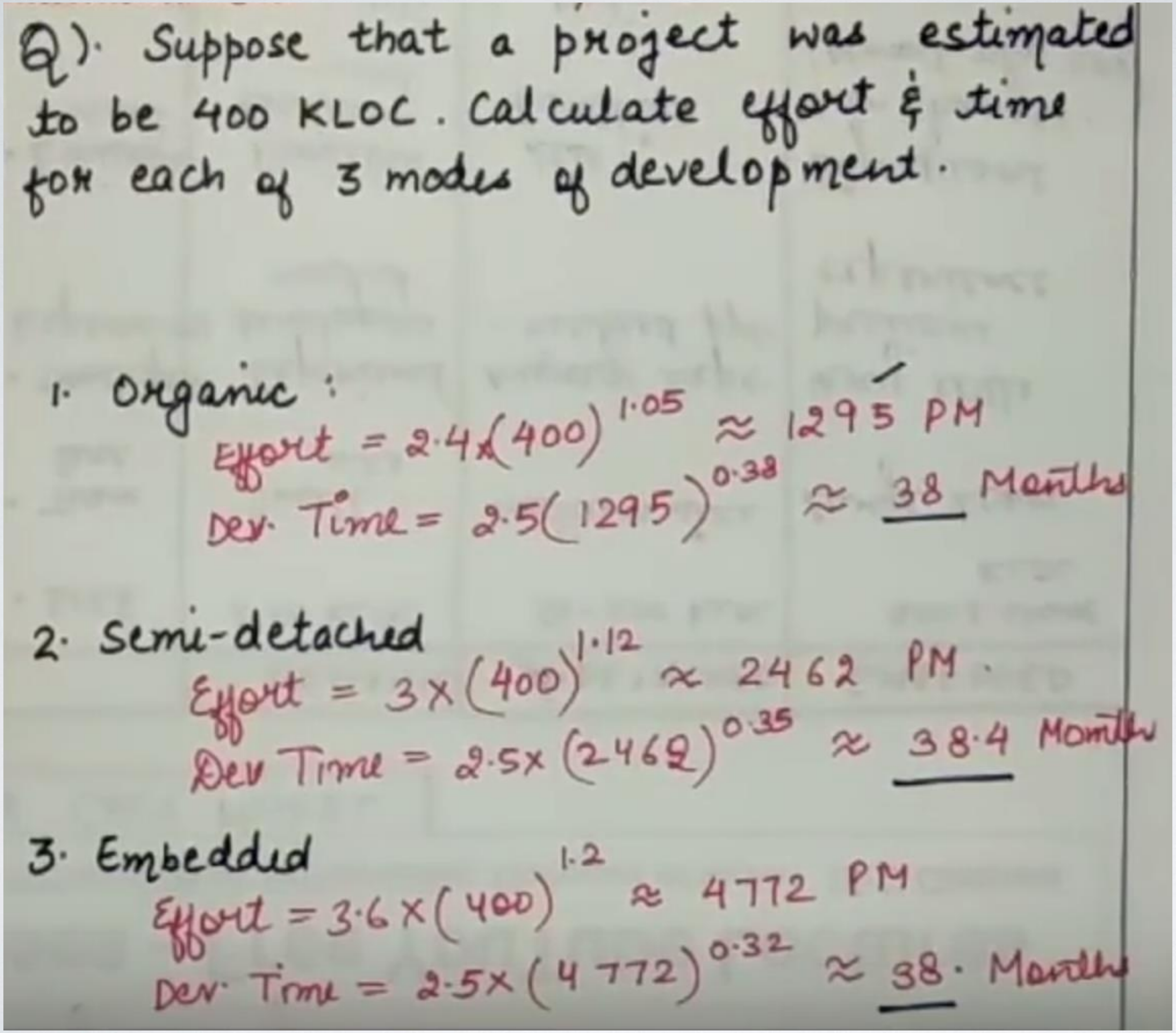
1· Effort $\rightarrow \boxed{a\,(KLOC)^b}$ person-month

2· Development Time $\rightarrow \boxed{c\,(Effort)^d}$ months

3· Average Staff Size $\rightarrow \boxed{\frac{Effort}{Dev \cdot Time}}$ persons

4· Productivity $\rightarrow \boxed{\frac{KLOC}{Effort}}$ KLOC/P·M

Software Product Type	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



Q). Suppose that a project was estimated to be 400 KLOC. Calculate effort & time for each of 3 modes of development.

1. Organic :

Effort $= 2.4 \times (400)^{1.05} \approx 1295$ PM

Dev. Time $= 2.5(1295)^{0.38} \approx \underline{38}$ Months

2. Semi-detached

Effort $= 3 \times (400)^{1.12} \approx 2462$ PM.

Dev Time $= 2.5 \times (2462)^{0.35} \approx \underline{38.4}$ Months

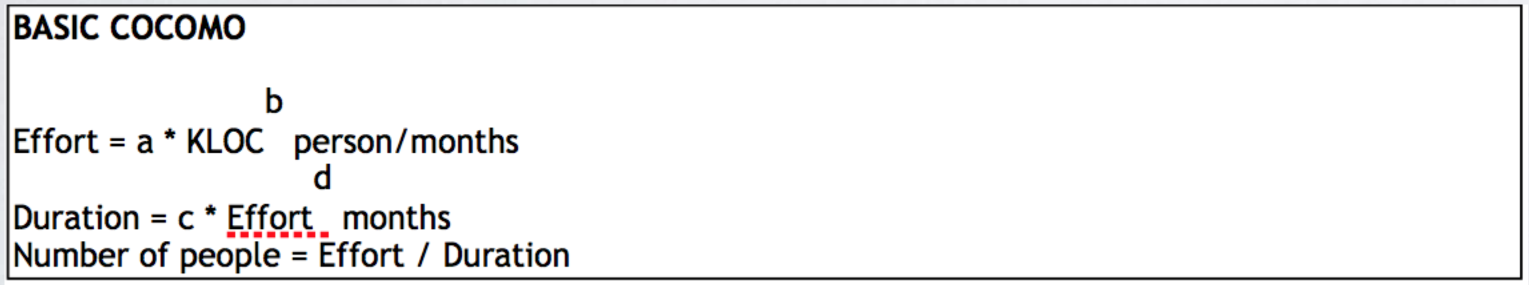
3. Embedded

Effort $= 3.6 \times (400)^{1.2} \approx 4772$ PM

Dev. Time $= 2.5 \times (4772)^{0.32} \approx \underline{38}$. Months

CLASS WORK

Using the Basic COCOMO 81 model (see the tables and formulae below), calculate the effort required (in person-months), the overall development time, and the number of personnel required for each product described below.

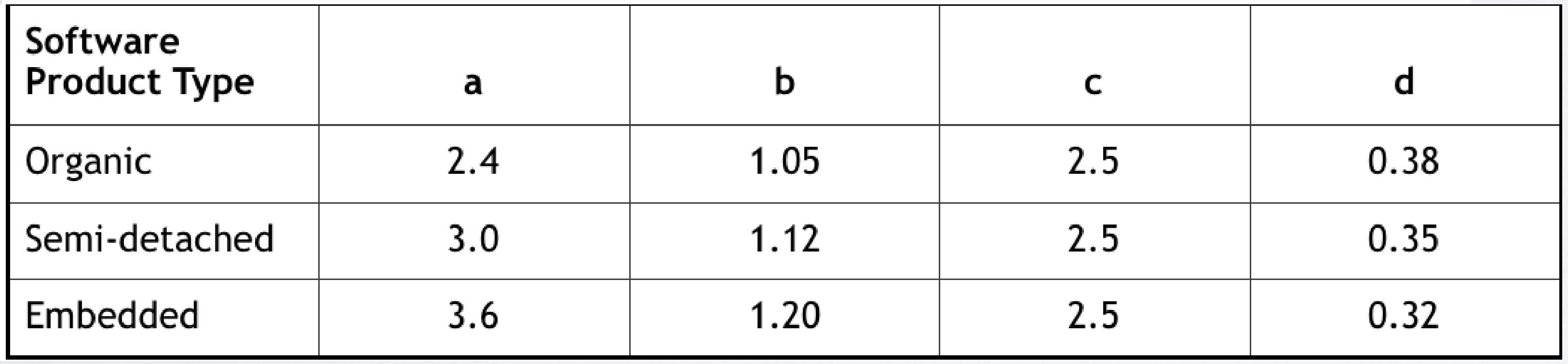


BASIC COCOMO

Effort = a * $KLOC^{b}$ person/months

Duration = c * $Effort^{d}$ months

Number of people = Effort / Duration



Software Product Type	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

Product 1*:* A semi-detached mode product delivering 75,000 lines of code.
Product 2: An embedded mode product delivering 75,000 lines of code.
Product 3*:* An organic mode product delivering 75,000 lines of code.

Briefly comment on your estimations for the above products

CLASS WORK

- There is a *highly complex, real-time system software project. The software size is 128 KLOC.* Using the Basic COCOMO model to **estimate the effort (person-months), duration time and number of personnel required.**

CLASS WORK

- Calculate using the **COCOMO model, the effort required in terms of man-months, the maximum time duration and the number of developers required for a project, whose RFP**(Request for Proposal) you just got. The initial outline of the project:

The project requires the developer to develop an E-Commerce system for a Toy Store located at Kathmandu. The store wishes to put various items online so that the potential customers outside Kathmandu can also see and buy the toys. The store primarily requires the developer to introduce online payment system, something which is quite new to the Nepalese market and something which you haven’t had an opportunity to implement in the previous ventures you’ve worked in. The estimated size of the project is **1200 KLOC.**