COURSE NAME – ADAPTIVE SOFTWARE ENGINEERING

COURSE CODE – 23CI2001

SESSION-26

TOPIC:

PROJECT PLANNING ESTIMATION

AIM OF THE SESSION

To familiarize students with the basic concept of Project planning and estimation

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

This Session is designed to:

1.Describe Project estimation
2Explain Function Point Analysis
3.Describe COCOMO Software Estimation Model

LEARNING OUTCOMES

At the end of this session, you should be able to:

1 Define project estimation
2. **Demonstrate** the order in which the following are estimated using the COCOMO estimate technique, use a schematic diagram and a relevant example: Cost, effort, length of time, and size .

AGENDA

- Project estimation
- Lines of code
- Function point analysis
- **COCOMO MODEL**

PROJECT ESTIMATION

Estimation determines how much money, effort, resources, and time will be required to build a specific system or product. Project estimation may involve the following:

- **Software size estimation**
- **Effort estimation**
- **Time estimation**
- **Cost estimation**

Software size estimation

It helps the project manager to predict the effort and time which will be needed to build the project. The following measures are used for size estimation:

- Lines of Code
- Number of entities in ER diagram
- Total number of processes in detailed data flow diagram
- Function points

Lines of Code

LOC count the total number of lines of source code in a project. The units of LOC are:

- KLOC- Thousand lines of code
- NLOC- Non comment lines of code
- KDSI- Thousands of delivered source instruction

Advantages

- Universally accepted and is used in many models like COCOMO.
- Estimation is closer to developer's perspective.
- Simple to use.

Disadvantages

Different programming languages contains different number of lines.

No proper industry standard exist for this technique.

It is difficult to estimate the size using this technique in early stages of project.

TOTAL NUMBER OF PROCESSES IN DETAILED DATA FLOW DIAGRAM

Data Flow Diagram(DFD) represents the functional view of a software. The model depicts the main processes/functions involved in software and flow of data between them. Utilization of number of functions in DFD to predict software size. Already existing processes of similar type are studied and used to estimate the size of the process. Sum of the estimated size of each process gives the final estimated size.

Advantages

- It is independent of programming language.
- Each major processes can be decomposed into smaller processes. This will increase the accuracy of estimation

Disadvantages

- Studying similar kind of processes to estimate size takes additional time and effort.
- All software projects are not required to construction the DFD.

FUNCTION POINT ANALYSIS

In this method, the number and type of functions supported by the software are utilized to find FPC (function point count). The steps in function point analysis are:

1. **Count the number of functions of each proposed type**
2. **Compute the Unadjusted Function Points (UFP)**
3. **Find Total Degree of Influence (TDI)**
4. **Compute Value Adjustment Factor (VAF)**
5. **Find the Function Point Count (FPC)**

Function Point Analysis: Continue

(1) Count the number of functions of each proposed type

Find the number of functions belonging to the following types:

- **External Inputs:** Functions related to data entering the system.
- **External outputs:** Functions related to data exiting the system.
- **External Inquiries:** They leads to data retrieval from system but don't change the system.
- **Internal Files:** Logical files maintained within the system. Log files are not included here.
- **External interface Files:** These are logical files for other applications which are used by our system.

(2) Compute the Unadjusted Function Points (UFP):

Categorize each of the five function types as **simple, average or complex** based on their complexity. Multiply count of each function type with its weighting factor and find the weighted sum.

- Weighting factors for each type based on their complexity

Function type	Simple	Average	Complex
External Inputs:	3	4	6
External outputs	4	5	7
External Inquiries	3	4	6
Internal Files	7	10	15
External interface Files	5	7	10

(3) FIND TOTAL DEGREE OF INFLUENCE (TDI)

Use the ’14 general characteristics’ of a system to find the degree of influence of each of them. The sum of all 14 degrees of influences will give the TDI. The range of TDI is 0 to 70. The 14 general characteristics are: **Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, On-Line Data Entry, End-user Efficiency, Online Update, Complex Processing Reusability, Installation Ease, Operational Ease, Multiple Sites and Facilitate Change.** Each of above characteristics is evaluated on a scale of 0-5.

(4) Compute Value Adjustment Factor (VAF):

Use the following formula to calculate VAF.

$$VAF = (TDI * 0.01) + 0.65$$

(5) Find the Function Point Count:

Use the following formula to calculate FPC.

$$FPC = UFP * VAF$$

Advantages

- It can be easily used in the early stages of project planning.
- It is independent on the programming language.
- It can be used to compare different projects even if they use different technologies(database, language etc).

Disadvantages

- It is not good for real time systems and embedded systems.
- Many cost estimation models like COCOMO uses LOC and hence FPC must be converted to LOC.

COCOMO MODEL
(CONSTRUCTIVE COST MODEL)

Introduction

- It is one of the most widely used for software estimation model in the world
- It was developed by Berry Boehm in 1981.
- COCOMO model predicts the **efforts** and **schedule** for a software development based on the input related to size of the software and a numbers of the cost drivers that effects the productivity.

COCOMO Models

Based on the **complexity** of the project, COCOMO has three following models:

- Basic COCOMO model
- Intermediate COCOMO model
- Detailed COCOMO model

THE DEVELOPMENT MODES: PROJECT CHARACTERISTICS

❑ Organic Mode

- Relatively **small**, simple software projects
- **Small teams** with **good** application **experience** work to a set of less than rigid requirements
- **Similar** to the previously developed projects relatively **small** and requires little innovation

❑ Semi-detached Mode

Intermediate (in size and complexity) software projects in which teams with **mixed experience** levels must meet a mix of rigid and less than rigid requirements.

❑EMBEDDED MODE

Software Projects That Must Be Developed Within A Set Of **Tight Hardware**, **Software**, And **Operational Constraints**.

❑Er

The Development Modes: Project Characteristics

Mode	Project Size	Nature of Project	Innovation	Deadline
Organic	Typically 2-50 KLOC	Small size project, Experienced developers.	Little	Not Tight
Semi Detached	Typically 50-300KLOC	Medium size project and team.	Medium	Medium
Embedded	Typically over 300KLOC	Large project, Real-time systems	Significant	Tight

Assumptions:

- Primary cost driver is the number of **Delivered Source Instructions** (DSI) / **Delivered Line Of Code** developed by the project.
- COCOMO estimation assume that the project will enjoy **good management** by both the developer and the customer.
- Assume the requirements specification is **not substantially changed** after the plans and requirements phase.

BASIC COCOMO

- Basic COCOMO is good for **quick, early, rough** model to **estimate the software efforts and costs.**
- It does not account for differences in **hardware constraints, personnel quality and experience**, **use of modern tools and techniques**, and other **project attributes** known to have a significant influence on software costs, which **limits its accuracy.**

BASIC COCOMO MODEL: FORMULA

1. **E= $a(KLOC\ or\ KDSI)^b$PM**

D= $c(E)^d$Months

P= E/D

E: The effort applied in person-months.

D: The development time in chronological months.

KLOC/KDSI: the estimated number of delivered lines of code for the project (expressed in thousands).

P: The number of people required.

Note: The effort and the duration estimations obtained using the COCOMO model are called as **nominal effort estimate and nominal duration estimate**.

- Coefficient a, b, c, and d:

Software project	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

Basic COCOMO Model: Equation

Mode	Effort	Schedule
Organic	$E=2.4*(KDSI)^{1.05}$	$TDEV=2.5*(E)^{0.38}$
Semidetached	$E=3.0*(KDSI)^{1.12}$	$TDEV=2.5*(E)^{0.35}$
Embedded	$E=3.6*(KDSI)^{1.20}$	$TDEV=2.5*(E)^{0.32}$

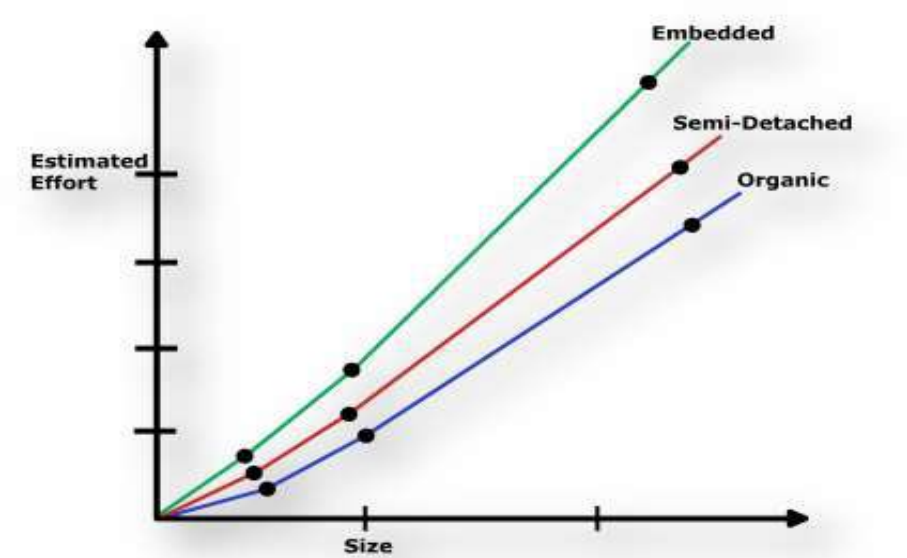

Fig. Effort Verses Product size

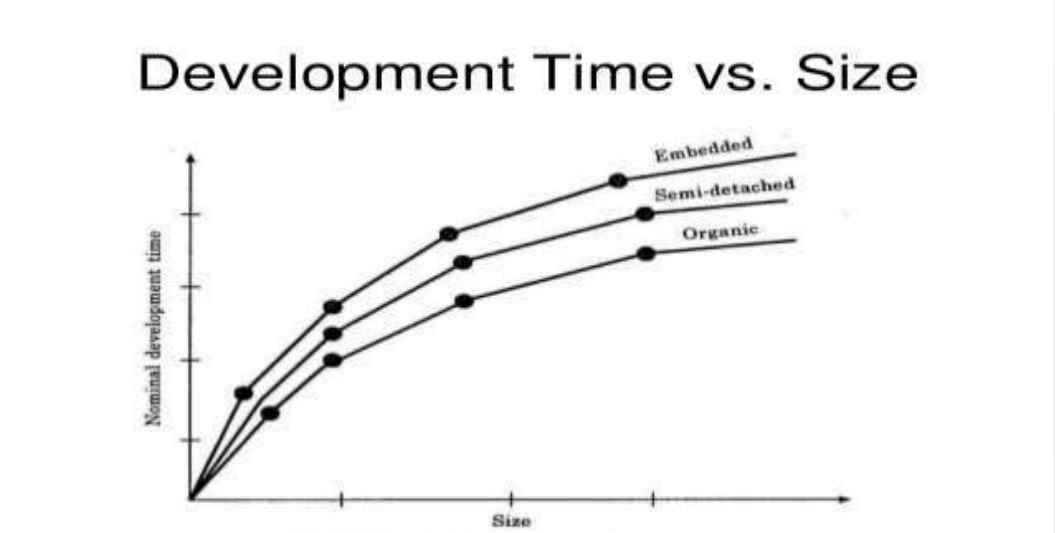

FIGURE 3.5 Development time vs. size.

- **Problem-1:** Assume that the size of an organic type software product has been estimated to be 32,000 lines of source code. Assume that the average salary of software engineers be Rs. 15,000/- per month. Determine the effort required to develop the software product and the nominal development time.
- **Solution:** As per the basic COCOMO estimation formula for organic software:

  Effort=$2.4*(32)^{1.05}$
- = 91 PM
-

  Nominal development time = $2.5 * (91)^{0.38}$ = 14 months

  Cost required to develop the product = 14 * 15,000
  = Rs. 210,000/-

PROBLEM 2: Suppose That A Project Was Estimated To Be 400 KLOC. Calculate The Effort And Development Time For Each Of The Three Modes I.E. Organic, Semidetached And Embedded.

- **Solution:** Estimated size of the project = 400 KLOC
- **1. Organic Mode**
- E = 2.4 (400)1.05 = 1295.31 PM
- D = 2.5 (1295.31)0.38 = 38.07 M
- **2. Semi detached Mode**
- E = 3.0 (400)1.12 = 2462.79 PM
- D = 2.5 (2462.79)0.35 = 38.45 M
- **3. Embedded Mode**
- E = 3.6 (400)1.20 = 4772.81 PM
- D = 2.5(4772.81)0.32 =37.59M

INTERMEDIATE COCOMO MODEL

- The basic COCOMO model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software system.
- However, in reality, no system's effort and schedule can be solely calculated on the basis of **Lines of Code**. For that, various other factors such as **reliability, Experience**, **Capability** are known as **Cost Drivers which can not be measured using LOC.**
- The Intermediate Model utilizes 15 such drivers for cost estimation.

EQUATIONS FOR INTERMEDIATE COCOMO MODEL

- **E=** $a(KLOC\ or\ KDSI)^b * EAF$
- **D=** $c(E)^d$
- **P=** E/D
- The project manager is to rate these 15 different parameters for a particular project on six point rating scale (**Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH), Extra High (EH)**). Then, based on these ratings, appropriate cost driver values are taken from the above table. These 15 values are then multiplied to calculate the **EAF (Effort Adjustment Factor).**

VALUES OF COEFFICIENT A, B, C, AND D FOR INTERMEDIATE COCOMO MODEL

Software project	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

SELF-ASSESSMENT QUESTIONS

- **1. Describe project estimation.**
- **2. Differentiate between FP and LOC**
- **3. Summarize the** COCOMO **Software Estimation Model**

REFERENCES FOR FURTHER LEARNING OF THE SESSION

TEXTBOOKS:

1. Roger S.Pressman, "Software Engineering – A Practitioner's Approach" 7th Edition, Mc Graw Hill,(2014).
2. Ian Sommerville, "Software Engineering", Tenth Edition, Pearson Education, (2015).
3. Agile Software Development Ecosystems, Jim Highsmith, Addison Wesley; ISBN: 0201760436; 1st edition

Reference Book

Agile Modelling: Effective Practices for Extreme Programming and the Unified Process Scott Amber John Wiley & Sons; ISBN: 0471202827; 1st edition.

WEB REFERNCES/MOOCS:

https://www.digite.com/kanban/what-is-kanban/
http://www.scaledagileframework.com
https://www.guru99.com/test-driven-development.html
https://junit.org/junit5/

THANK YOU

(DEEMED TO BE UNIVERSITY)

Team – ADAPTIVE SOFTWARE ENGINEERING