Terna Engineering College

Computer Engineering Department

Program: Sem VI

Course: Software Engineering Lab

LAB Manual

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No.09

A.1 Aim:

Estimation of Project Metrics using COCOMO.

A.2 Prerequisite:

1. Preliminary requirements
2. Knowledge about how to estimate cost and time

A.3 Outcome:

After successful completion of this experiment, students will be able to Categorize project using COCOMO and estimate the effort and development time required for a project.

A.4 Theory:

- **Project Estimation Techniques**

A software project is not just about writing a few hundred lines of source code to achieve a particular objective. The scope of a software project is comparatively quite large, and such a project could take several years to complete. However, the phrase "quite large" could only give some (possibly vague) qualitative information. As in any other science and engineering discipline, one would be interested to measure how complex a project is. One of the major activities of the project planning phase, therefore, is to estimate various project parameters to make proper decisions. Some important project parameters that are estimated include:

Project size: What would be the size of the code written say, in several lines, files, modules?

Cost: How much would it cost to develop software? Software may be just pieces of code, but one has to pay the managers, developers, and other project personnel.

Duration: How long would it be before the software is delivered to the clients?

Effort: How much effort from the team members would be required to create the software?

COCOMO: COCOMO (Constructive Cost Model)

- COCOMO was proposed by Boehm. According to him, there could be three categories of software projects: organic, semidetached, and embedded. The classification is done considering the characteristics of the software, the development team and the environment. These product classes typically correspond to application, utility and system programs, respectively. Data processing programs could be considered application programs. Compilers, linkers, are examples of utility programs. Operating systems, real-time system programs are examples of system programs. One could easily apprehend that it would take much more time and effort to develop an OS than an attendance management system.
- The concept of organic, semidetached, and embedded systems are described below.

  ✔ **Organic**: A development project is said to be of organic type if The project deals with developing a well-understood application, The development team is small, The team members have prior experience in working with similar types of projects

  ✔ **Semidetached**: A development project can be categorized as semidetached type, if The team consists of some experienced as well as inexperienced staff, Team members may have some experience on the type of system to be developed

✔ **Embedded:** Embedded type of development project are those, which

Aims to develop a software strongly related to machine hardware Team size is usually large

- Boehm suggested that estimation of project parameters should be done through three stages: Basic COCOMO, Intermediate COCOMO, and Complete COCOMO.

✔ **Basic COCOMO Model:**

The basic COCOMO model helps to obtain a rough estimate of the project parameters. It estimates the effort and time required for development in the following way:

$Effort = a * (KLOC)^b$ PM

$Tdev = 2.5 * (Effort)^c$ Months

Where KLOC is the estimated size of the software expressed in Kilo Delivered Source Instructions

a, b, c are constants determined by the category of software project

Effort denotes the total effort required for the software development, expressed in person-months (PMs)

Tdev denotes the estimated time required to develop the software (expressed in months)

The value of the constants a, b, c is given below:

Software project	a	B	c
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

✔ Intermediate COCOMO Model:

The basic COCOMO model considers that effort and development time depends only on the size of the software. However, in real life, many other project parameters influence the development process. The intermediate COCOMO consider those other factors by defining a set of 15 cost drivers (multipliers) as shown in the table below [i]. Thus, any project that makes use of modern programming practices would have lower estimates in terms of effort and cost. Each of the 15 such attributes can be rated on a six-point scale ranging from "very low" to "extra high" in their relative order of importance. Each attribute has an effort multiplier fixed as per the rating. The product of effort multipliers of all the 15 attributes gives the Effort Adjustment Factor (EAF).

Cost drivers for Intermediate COCOMO

(Source: http://en.wikipedia.org/wiki/COCOMO)

Cost Drivers Ratings						
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints				1.00	1.11	1.30	1.66
Memory constraints		1.00	1.06	1.21	1.56	
The volatility of the virtual machine environment					0.87	1.00	1.15	1.30
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		

Programming language experience	1.14	1.07	1.00	0.95		

Project attributes

Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

EAF is used to refine the estimates obtained by basic COCOMO as follows:

Effort|corrected = Effort * EAF

Tdev|corrected = 2.5 * (Effort| corrected)

✔ **Complete COCOMO Model (Advanced COCOMO):**

Both the basic and intermediate COCOMO models consider software to be a single homogeneous entity -- an assumption, which is rarely true. Many real-life applications are made up of several smaller sub-systems. (One might not even develop all the subsystems -- just use the available services). The complete COCOMO model takes these factors into account to provide a far more accurate estimate of project metrics.

To illustrate this, consider a very popular distributed application: the ticket booking system of the Indian Railways. There are computerized ticket counters in most of the railway stations of our country. Tickets can be booked/cancelled from any such counter. Reservations for future tickets, cancellation of reserved tickets could also be performed. On a high level, the ticket booking system has three main components:

Database, Graphical User Interface (GUI), Networking facilities

Among these, the development of the GUI is considered an organic project type; the database module could be considered as a semi-detached software. The networking module can be considered as embedded software. To obtain a realistic cost, one should estimate the costs for each component separately, and then add it up.

Advantages of COCOMO:

COCOMO is a simple model and should help one to understand the concept of project metrics estimation.

Drawbacks of COCOMO:

COCOMO uses KLOC, which is not a proper measure of a program's size. Indeed, estimating the size of a software is a difficult task, and any slight miscalculation could cause a large deviation in subsequent project estimates. Moreover, COCOMO was proposed in 1981 keeping the waterfall model of the project life cycle in mind [2]. It fails to address other popular approaches like prototype, incremental, spiral, agile models. Moreover, in the present-day, a software project may not necessarily consist of coding of every bit of functionality. Rather, existing software components are often used and glued together towards the development of new software. COCOMO is not suitable in such cases.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Blackboard access available)

Roll No. 50	**Name:** AMEY THAKUR
Class: Comps TE B	**Batch:** B3
Date of Experiment: 28/03/2021	**Date of Submission:** 28/03/2021
Grade:	

B.1 Question of Curiosity

(To be answered by student based on the practical performed and learning/observations)

1. **According to the COCOMO model, a project can be categorized into**
   - 3 types
   - 5 types
   - 5 types
   - no such categorization

Ans: 3 types

2. **In the Intermediate COCOMO model, Effort Adjustment Factor (EAF) is derived from the effort multipliers by**
   - Adding them
   - Multiplying them
   - Taking their weighted average
   - Considering their maximum

Ans: Multiplying them

3. **Project metrics are estimated during which phase?**
   - Feasibility study
   - Planning
   - Design
   - Development

Ans: Planning

4. Suppose you are developing a software product in the organic model. You have estimated the size of the product to be 70695 lines of code, **compute effort and development time**. Assuming a cost of 25,000 person month **calculate the total cost of the product**
(constant aa = 2.4. bb = 1.05, bc = 2.5, bd = 0.38)

Ans:

AMEY B 50 Amey

$\therefore E = aa \ KLOC^{bb}$

$= 2.4 \ (70695)^{1.05}$

$= 210$

$\therefore E = 210$ Person - Months

$\therefore D = bc \cdot E^{bd}$

$= 2.5 \ (210)^{0.38}$

$= 19.07$

$\therefore D = 19.07$ Months

$\therefore$ Estimated cost $= 210 \times 25000$

$= 5250000$ Rs.

B.2 Conclusion:

(Students must write the conclusion)

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.