Estimating Efforts and Cost

(Function Point & COCOMO Method)

COCOMO Model

The COCOMO (COnstructive COst MOdel) model is an empirical and algorithmic based model that was derived by collecting data from a large number of software projects. These data were analysed to discover formulae that were the best fit to the observations. These formulae link the size of the system and product, project and team factors to the effort to develop the system.

COnstructive COst MOdel. Boehm's hierarchy of models takes the following form:

- ***Model 1*:** The basic COCOMO model computes software development effort (and cost) as a function of program size expressed in estimated lines of code (LOC).
- ***Model 2*:** The Intermediate COCOMO model computes software development effort as a finction of program size and set of "cost divers" that include subjective assessments of product, hardware, personnel, and project attributes.
- ***Model 3*:** The Advanced COCOMO model incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

There are **four steps** that the user needs to follow in **COCOMO**, which are:
(http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/gamel/cocomo.html)

Step 1: The user should fill the Information Domain table in which he/she can get the **Count Total (CT)** which will be used in the FP equation. The Information Domain are defined in the following manner:

- ***Number of user inputs*** : Each user input that provides distinct application oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.

- ***Number of user outputs*** : Each user output that provides application-oriented information to the user is counted. In this context output refers to reports, screens, error messages, and so on. Individual data items within a report are not counted.
- ***Number of user inquiries :*** An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.
- ***Number of files :*** Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file), is counted.
- ***Number of external interfaces:*** All machine readable interfaces (e.g., data files on tape or disk) that are used to transmit information to another system are counted.

Once the above data have been collected, a complexity value is associated with each count. Once all the information are entered, the **Count Total (CT)** is calculated. Following is an example of this step.

Information Domain Values

Measurement Parameter	Count		Simple ○	Average ◉	Complex ○		Total
Number of user inputs	3	X	3	4	6	=	12.00
Number of user outputs	5	X	4	5	7	=	25.00
Number of user inquiries	2	X	3	4	6	=	8.00
Number of files	1	X	7	10	15	=	10.00
Number of external interfaces	1	X	5	7	10	=	7.00
Count_Total (CT)							62.00

Step 2: The end user should calculate the "Complexity/ Value Adjustment Factor" ($\sum$ Fi where i = 1 to 14). The user will give a value between 0 to 5. Once Step 1 and Step 2 are calculated, then the end user can calculated the **Function Points (FP)** which is:

$$FP = CT * [0.65 + 0.01 * \sum Fi]$$

Complexity Weighting Factors

// heading of the second table Rate each factor on a scale of 0 to 5:

(0 = No influence, 1 = Incidental, 2 = Moderate, 3 = Average, 4 = Significant, 5 = Essential):

Question	0	1	2	3	4	5
1. Does the system require reliable backup and recovery?	○	○	◉	○	○	○
2. Are data communications required?	○	○	○	◉	○	○
3. Are there distributed processing functions?	○	○	○	◉	○	○
4. Is performance critical?	○	○	○	○	◉	○
5. Will the system run in an existing, heavily utilized operational environment?	○	○	○	○	◉	○
6. Does the system require on-line data entry?	○	◉	○	○	○	○
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?	◉	○	○	○	○	○
8. Are the master file updated on-line?	◉	○	○	○	○	○
9. Are the inputs, outputs, files, or inquiries complex?	○	○	◉	○	○	○
10. Is the internal processing complex?	◉	○	○	○	○	○
11. In the code designed to be reusable?	○	○	◉	○	○	○
12. Are conversion and installation included in the design?	○	○	○	○	◉	○
13. Is the system designed for multiple installations in different organizations?	○	○	◉	○	○	○
14. Is the application designed to facilitate change and ease of use by the user?	○	○	◉	○	○	○
Total 29.00						

Show Total of weighting Factor

The Function Points is: Show Function Points 58.28

Step 3: The end user should select a programming language from the Table found in step 3 on the main page that provides a rough estimate of the average number of lines of code required to build one function point in various programming languages. Once the programming language is selected, then the end user can calculate the Line Of Code (LOC).

Thus the required **KLOC** is determined for the software project.

Programming Language	LOC/FP (average)	Select
Assembly Language	320	○
C	128	○
COBOL	105	○
Fortran	105	○
Pascal	90	○
Ada	70	○
Object-Oriented Languages	30	◉
Fourth Generation Languages (4GLs)	20	○
Code Generators	15	○
Spreadsheets	6	○
Graphical Languages (icons)	4	○

LOC/F P: Show LOC/FP 1748.40

Step 4: This is the ***final step*** of the basic COCOMO model. Here we will find **Efforts** and **Duration** of the project. The end user has to select one of the three (3) types of modes, which are organic, semi-detached, and embedded.

Organic Model: Relatively small, simple software projects in which a small teams with good application experience work to a set of less than rigid requirement. The equation for the Effort (E) and Development time (D) for this model are:

E = 2.4 * (KLOC)^1.05 D = 2.5 * (E)^0.38

Semi-Detached Model: An intermediate (in size and complexity) software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements. The equation for the Effort (E) and Development time (D) for this model are:

E = 3.0 * (KLOC)^1.12 D = 2.5 * (E)^0.35

Embedded Model: A software project that must be developed within a set of tight hardware, software and operational constraints. The equation for the Effort (E) and Development time (D) for this model are:

E = 3.6 * (KLOC)^1.20 D = 2.5 * (E)^0.32

Once the end user selects his/her model, he/she calculates the (E)ffort and the (D)evelopment time.

Software Project	a_b	b_b	c_b	d_b	Select
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	○

Calculate Effort and Duration

Effort (E) = $a_b(KLOC)^{b}_{b}$ = 5.61 Duration (D) = $c_b(E)^{d}_{b}$ = 4.57

Note: **Effort** is in ***person-month.***

Duration is in ***month***.

The **relationship between** the number of **staff** working on a project, the total effort required and the development **time** is **not linear**. As the number of staff increases, more effort may be needed. The reason for this is that people spend more time communicating and defining interfaces between the parts of the system developed by other people. Doubling the number of staff (for example) therefore does not mean that the duration of the project will be halved.

- Software project is split into a number of separate activities. The software project planning concentrated on ways to represent these activities, their dependencies and the allocation of people to carry out these tasks.

- As a software project manager you need to estimate of effort and time with the project activities. Estimation involves answering the following questions:
    1. How much effort is required to complete each activity?
    2. How much calendar time is needed to complete each activity?
    3. What is the total cost of each activity?

- There are three parameters involved in computing the **Total Cost** of a **software development project:**

    1. **Hardware and software costs including maintenance**
    2. **Travel and training costs**
    3. **Effort costs** (the costs of paying software engineers)

For most projects, the **dominant cost is the effort cost**. Computers that are powerful enough for software development are relatively cheap.

Effort costs are not just the salaries of the software engineers who are involved in the project. Organization's compute effort costs in terms of overhead costs where they take the total cost of running the organization and divide this by the number of productive staff.

Therefore, the following costs are all ***part of the total effort cost***:

1. Costs of providing, heating and lighting office space
2. Costs of support staff such as accountants, administrators, system
3. managers,
4. cleaners and technicians
5. Costs of networking and communications
6. Costs of central facilities such as a library or recreational facilities
7. Costs of Social Security and employee benefits such as pensions and health insurance.

This ***overhead factor is usually at least twice the software engineer's salary***, depending on the size of the organization and its associated overheads. **Therefore, if a company pays a software engineer $90,000 per year, its total cost is at least $180,000 per year.**