overcomes an important shortcoming of the expert judgement technique in that the results can not unjustly be influenced by overly assertive and senior members.

3.7 COCOMO—A HEURISTIC ESTIMATION TECHNIQUE

COnstructive COst estimation MOdel (COCOMO) was proposed by Boehm [1981]. COCOMO prescribes a three stage process for project estimation. In the first stage, an initial estimate is arrived at. Over the next two stages, the initial estimate is refined to arrive at a more accurate estimate. COCOMO uses both single and multivariable estimation models at different stages of estimation.

The three stages of COCOMO estimation technique are—basic COCOMO, intermediate COCOMO, and complete COCOMO. We discuss these three stages of estimation in the following subsection.

3.7.1 Basic COCOMO Model

Boehm postulated that any software development project can be classified into one of the following three categories based on the development complexity—organic, semidetached, and embedded. Based on the category of a software development project, he gave different sets of formulas to estimate the effort and duration from the size estimate.

Three basic classes of software development projects

In order to classify a project into the identified categories, Boehm requires us to consider not only the characteristics of the product but also those of the development team and development environment. Roughly speaking, the three product development classes correspond to development of application, utility and system software. Normally, data processing programs[1] are considered to be application programs. Compilers, linkers, etc., are utility programs. Operating systems and real-time system programs, etc. are system programs. System programs interact directly with the hardware and programming complexities also arise out of the requirement for meeting timing constraints and concurrent processing of tasks.

Brooks [1975] states that utility programs are roughly three times as difficult to write as application programs and system programs are roughly three times as difficult as utility programs. Thus according to Brooks, the

relative levels of product development complexity for the three categories (application, utility and system programs) of products are 1:3:9.

Boehm's [1981] definitions of organic, semidetached, and embedded software are elaborated as follows:

Organic: We can classify a development project to be of organic type, if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.

Semidetached: A development project can be classify to be of semidetached type, if the development team consists of a mixture of experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.

Embedded: A development project is considered to be of embedded type, if the software being developed is strongly coupled to hardware, or if stringent regulations on the operational procedures exist. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.

Observe that in deciding the category of the development project, in addition to considering the characteristics of the product being developed, we need to consider the characteristics of the team members. Thus, a simple data processing program may be classified as semidetached, if the team members are inexperienced in the development of similar products.

For the three product categories, Boehm provides different sets of expressions to predict the effort (in units of person-months) and development time from the size estimation given in kilo lines of source code (KLSC). But, how much effort is one person-month?

> One person month is the effort an individual can typically put in a month. The person-month estimate implicitly takes into account the productivity losses that normally occur due to time lost in holidays, weekly offs, coffee breaks, etc.

What is a person-month?

Person-month (PM) is a popular unit for effort measurement.

> Person-month (PM) is considered to be an appropriate unit for measuring effort, because developers are typically assigned to a project for a certain number of months.

It should be carefully noted that an effort estimation of 100 PM does not imply that 100 persons should work for 1 month. Neither does it imply that 1 person should be employed for 100 months to complete the project. The effort estimation simply denotes the area under the person-month curve (see Figure 3.3) for the project. The plot in Figure 3.3 shows that different number of personnel may work at different points in the project development. The number of personnel working on the project usually increases or decreases by an integral number, resulting in the sharp edges in the plot. We shall elaborate in Section 3.9 how the exact number of persons to work at any time on the product development can be determined from the effort and duration estimates.

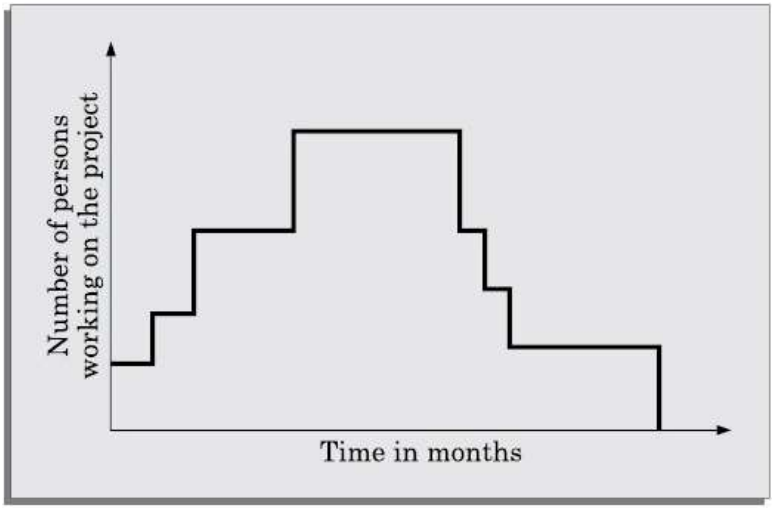

Figure 3.3: Person-month curve.

General form of the COCOMO expressions

The **basic COCOMO model** is a single variable heuristic model that gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by expressions of the following forms:

$$\text{Effort} = a_1 \times (\text{KLOC})a_2 \text{ PM}$$

$$\text{Tdev} = b_1 \times (\text{Effort})b_2 \text{ months}$$

where,

- KLOC is the estimated size of the software product expressed in Kilo Lines Of Code.
- a_1, a_2, b_1, b_2 are constants for each category of software product.
- Tdev 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).

According to Boehm, every line of source text should be calculated as one LOC irrespective of the actual number of instructions on that line. Thus, if a single instruction spans several lines (say n lines), it is considered to be nLOC. The values of a_1, a_2, b_1, b_2 for different categories of products as given by Boehm [1981] are summarised below. He derived these values by examining historical data collected from a large number of actual projects.

Estimation of development effort: For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic : Effort = 2.4(KLOC)1.05 PM

Semi-detached : Effort = 3.0(KLOC)1.12 PM

Embedded : Effort = 3.6(KLOC)1.20 PM

Estimation of development time: For the three classes of software products, the formulas for estimating the development time based on the effort are given below:

Organic : Tdev = 2.5(Effort)0.38 Months

Semi-detached : Tdev = 2.5(Effort)0.35 Months

Embedded : Tdev = 2.5(Effort)0.32 Months

We can gain some insight into the basic COCOMO model, if we plot the estimated effort and duration values for different software sizes. Figure 3.4 shows the plots of estimated effort versus product size for different categories of software products.

Observations from the effort-size plot From Figure 3.4, we can observe that the effort is some what superlinear (that is, slope of the curve>1) in the size of the software product.

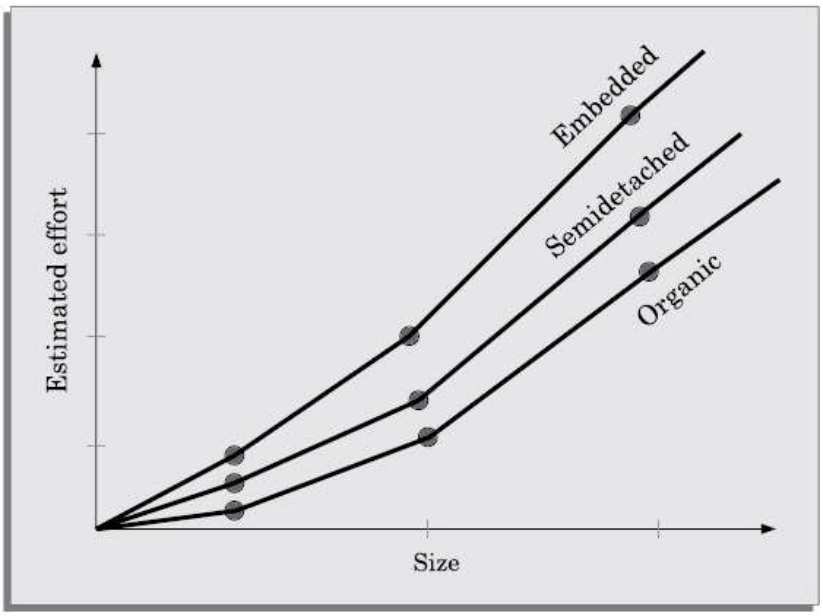

Figure 3.4: Effort versus product size.

This is because the exponent in the effort expression is more than 1. Thus, the effort required to develop a product increases rapidly with project size. However, observe that the increase in effort with size is not as bad as that was portrayed in Chapter 1. The reason for this is that COCOMO assumes that projects are carefully designed and developed by using software engineering principles.

Observations from the development time—size plot

The development time versus the product size in KLOC is plotted in Figure 3.5. From Figure 3.5, we can observe the following:

- The development time is a sublinear function of the size of the product. That is, when the size of the product increases by two times, the time to develop the product does not double but rises moderately. For example, to develop a product twice as large as a product of size 100KLOC, the increase in duration may only be 20 per cent. It may appear surprising that the duration curve does not increase superlinearly—one would normally expect the curves to behave similar to those in the effort-size plots. This apparent anomaly can be explained by the fact that COCOMO assumes that a project development is carried out not by a single person but by a team of developers.
- From Figure 3.5 we can observe that for a project of any given size, the

development time is roughly the same for all the three categories of products. For example, a 60 KLOC program can be developed in approximately 18 months, regardless of whether it is of organic, semi-detached, or embedded type. (Please verify this using the basic COCOMO formulas discussed in this section). However, according to the COCOMO formulas, embedded programs require much higher effort than either application or utility programs. We can interpret it to mean that there is more scope for parallel activities for system programs than those in utility or application programs.

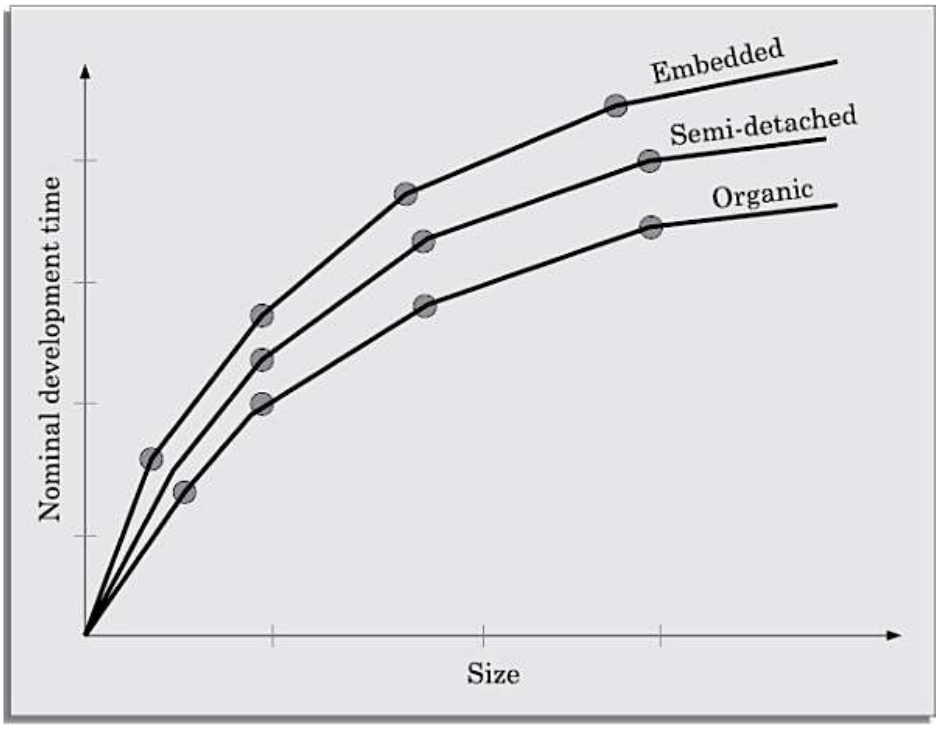

Figure 3.5: Development time versus size.

Cost estimation

From the effort estimation, project cost can be obtained by multiplying the estimated effort (in man-month) by the manpower cost per month. Implicit in this project cost computation is the assumption that the entire project cost is incurred on account of the manpower cost alone. However, in addition to manpower cost, a project would incur several other types of costs which we shall refer to as the overhead costs. The overhead costs would include the costs due to hardware and software required for the project and the company overheads for administration, office space,electricity, etc. Depending on the expected values of the overhead costs, the project manager has to suitably scale up the cost

arrived by using the COCOMO formula.

Implications of effort and duration estimate

An important implicit implication of the COCOMO estimates are that if you try to complete the project in a time shorter than the estimated duration, then the cost will increase drastically. But, if you complete the project over a longer period of time than that estimated, then there is almost no decrease in the estimated cost value. The reasons for this are discussed in Section 3.9. Thus, we can consider that the COCOMO effort and duration values to indicate the following.

> The effort and duration values computed by COCOMO are the values for completing the work in the shortest time without unduly increasing manpower cost.

Let us now elaborate the above statement. When a project team consists of a single member, the member would never be idle for want of work, but the project would take too long to complete. On the other hand, when there are too many members, the project would be completed in much shorter time, but often during the project duration some members would have to idle for want of work.

The project duration is as computed by the COCOMO model, all the developers remain busy with work during the entire development period. Whenever a project is to be completed in a time shorter than the duration estimated by using COCOMO, some idle time on the part of the developers would exist. Such idle times would result in increased development cost. An optimum sized team for a project is one in which any developer any time during development does not sit idle waiting for work, but at the same time consists of as many members as possible to reduce the development time. We can think of the duration given by COCOMO is called the as the optimal duration. It is called optimal duration, if the project is attempted to be completed in any shorter time, then the effort required would rise rapidly. This may appear as a paradox—after all, it is the same product that would be developed, though over a shorter time, then why should the effort required rise rapidly? This can be explained by the fact that for every product at any point during the project development, there is a limit on the number of parallel activities that can meaningfully be identified and carried out. Thus if more number of developers are deployed than the optimal size, some of the developers would have to idle, since at some point in development or other, it would not be possible to assign them any work at all. These idle times

would show up as higher effort and larger cost.

Staff-size estimation

Given the estimations for the project development effort and the nominal development time, can the required staffing level be determined by a simple division of the effort estimation by the duration estimation? The answer is "No". It will be a perfect recipe for project delays and cost overshoot. We examine the staffing problem in more detail in Section 3.9. From the discussions in Section 3.9, it would become clear that the simple division approach to obtain the staff size would be highly improper.

Example 3.2 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 a software developer is Rs. 15,000 per month. Determine the effort required to develop the software product, the nominal development time, and the cost to develop the product.

From the basic COCOMO estimation formula for organic software: Effort = $2.4 \times (32)1.05 = 91$ PM

Nominal development time = $2.5 \times (91)0.38 = 14$ months

Staff cost required to develop the product = $91 \times$ Rs. 15, 000 = Rs. 1,465,000

3.7.2 Intermediate COCOMO

The basic COCOMO model assumes that effort and development time are functions of the product size alone. However, a host of other project parameters besides the product size affect the effort as well as the time required to develop the product. For example the effort to develop a product would vary depending upon the sophistication of the development environment.

Therefore, in order to obtain an accurate estimation of the effort and project duration, the effect of all relevant parameters must be taken into account. The intermediate COCOMO model recognises this fact and refines the initial estimates.

The intermediate COCOMO model refines the initial estimate obtained using the basic COCOMO expressions by scaling the estimate up or down based on the evaluation of a set of attributes of software development.