responsibility of compiling the results and preparing the final estimate.

3.6 COCOMO—A HEURISTIC ESTIMATION TECHNIQUE

COCOMO (COnstructive COst estimation MOdel) was proposed by Boehm, 1981. Boehm postulated that any software development project can be classified into any one of the following three categories based on the development complexity: organic, semidetached, and embedded. In order to classify a product 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 classes correspond to application, utility and system programs, respectively. 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 typically involve meeting timing constraints and concurrent processing.

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 systems are elaborated as follows:

1. **Organic:** We can consider a development project to be of organic type, if the project deals with developing a well-understood application program, the size of the development

[1]A data processing program is one which processes large volumes of data using a simple algorithm. An example of a data processing application is a payroll software. A payroll software computes the salaries of the employees and prints cheques for them. In a payroll software, the algorithm for pay computation is fairly simple. The only complexity that arises while developing such a software product arises from the fact that the pay computation has to be done for a large number of employees.

team is reasonably small, and the team members are experienced in developing similar types of projects.

2. Semidetached: A development project can be considered 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.

3. Embedded: A development project is considered to be of embedded type, if the software being developed is strongly coupled to complex hardware, or if stringent regulations on the operational procedures exist.

Observe that Boehm in addition to considering the characteristics of the product being developed, considers the characteristics of the team members in deciding the category of the development project. 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 KLOC (Kilo Lines of Source Code). One person-month is the effort an individual can typically put in a month. This effort estimate takes into account the productivity losses that may occur due to lost time such as holidays, weekly offs, coffee breaks, etc.

Note that effort estimation is expressed in units of person-months (PM). 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. The person-month unit indicates the work done by one person working on the project for one month. It should be carefully noted that an effort estimation of 100 PM does not imply that 100 persons should work for 1 month. Neither nor does it imply that 1 person should be employed for 100 months. 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

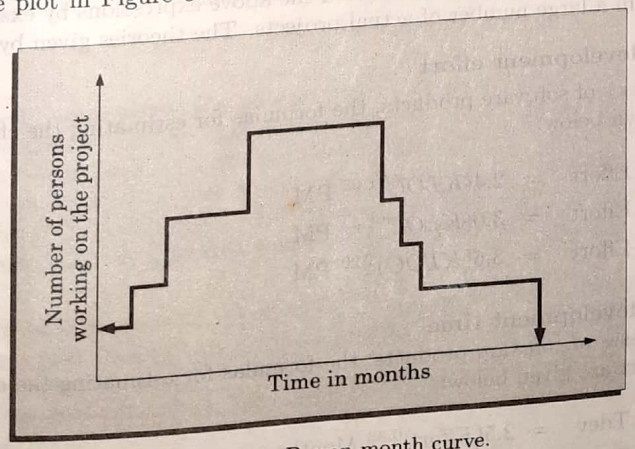

Figure 3.3: Person-month curve.

different point in the project development, as is typical in a practical industry scenario. The number of personnel working on the project usually increases and decreases by an integral

number, resulting in the sharp edges in the plot. We shall elaborate in Section 3.8 how the number of persons to work at any time on the product development is determined.

According to Boehm, software cost estimation should be done through three stages: basic COCOMO, intermediate COCOMO, and complete COCOMO. We discuss these stages as follows:

3.6.1 Basic COCOMO Model

The **basic COCOMO model** gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

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

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

Where

(a) KLOC is the estimated size of the software product expressed in Kilo Lines of Code,

(b) a_1, a_2, b_1, b_2 are constants for each category of software products,

(c) Tdev is the estimated time to develop the software, expressed in months,

(d) 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]. He derived the above expressions by examining historical data collected from a large number of actual projects. The theories given by Boehm were:

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

Semidetached : 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

Semidetached : 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 characteristics for different software sizes. Figure 3.4 shows a plot of estimated effort versus product size. From Figure 3.4, we can observe that the effort is somewhat superlinear (slope of the curve ≥ 1) in the size of the software product. 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.

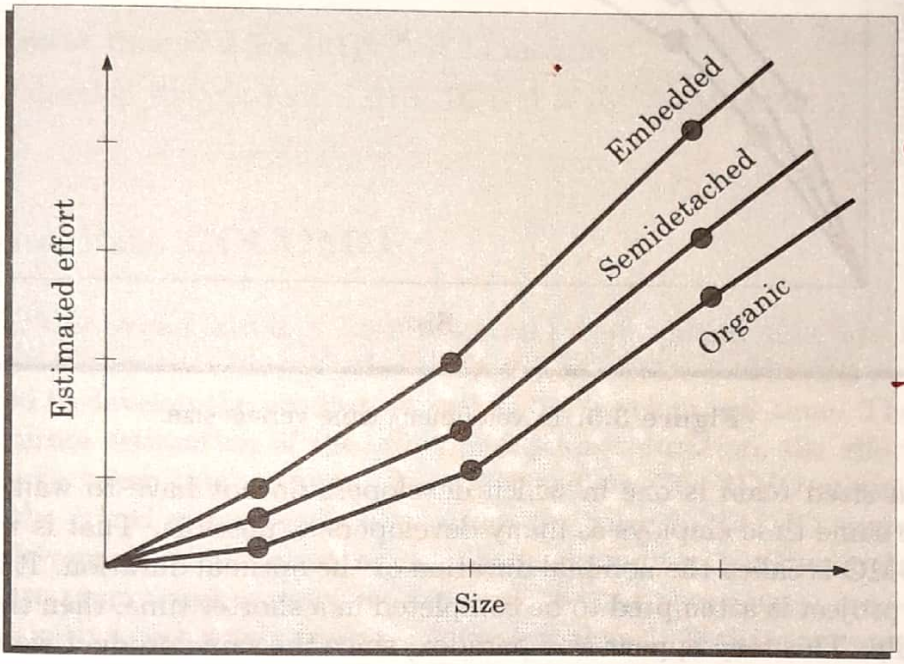

Figure 3.4: Effort versus product size.

The development time versus the product size in KLOC is plotted in Figure 3.5. From Figure 3.5, we can observe that 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. It may appear surprising that the duration curve does not increase superlinearly. The apparent anomaly can be explained by the fact that COCOMO assumes that a project is carried out not by a single person but by a team of developers.

It is important to note that the effort and duration estimations obtained using the COCOMO model imply 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 3.8. Thus, we can consider that the computed effort and duration values to indicate the following.

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

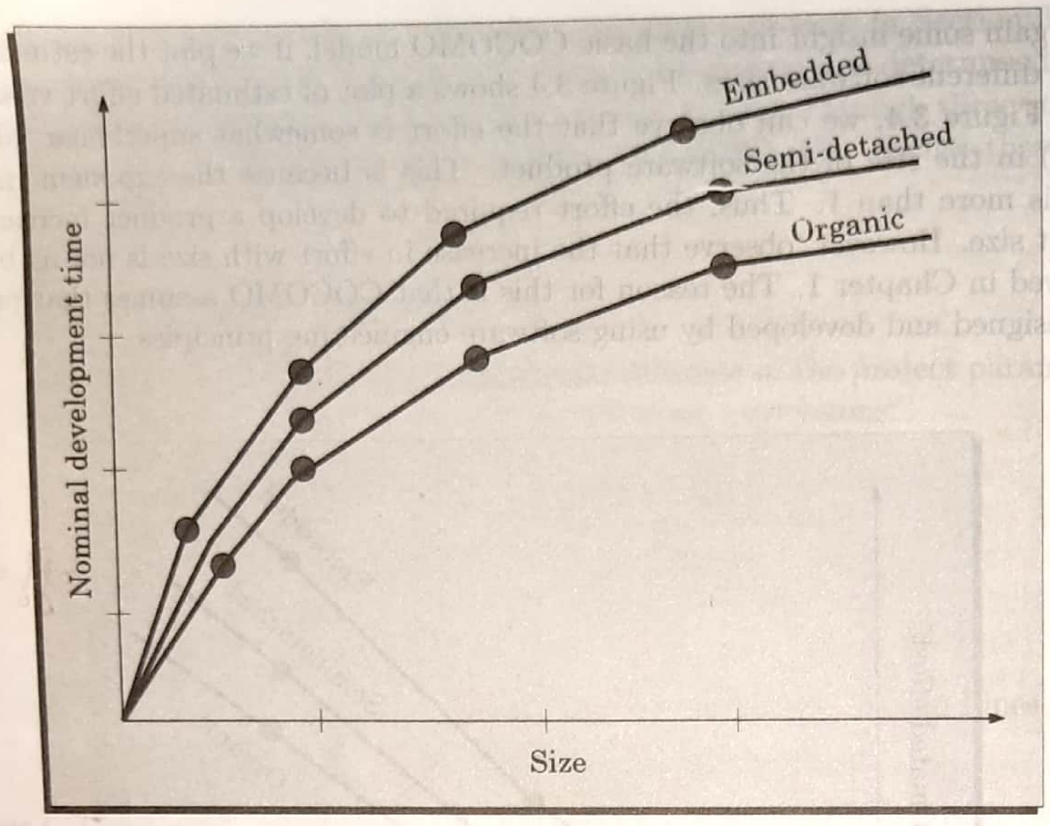

Figure 3.5: Development time versus size.

An optimum sized team is one in which developers do not have to wait idle waiting for work but at the same time employs as many developers as possible. That is why the duration given by COCOMO is called the nominal duration or the optimal duration. It is called optimal duration, if the project is attempted to be completed in a shorter time, then the effort required would rise rapidly. This may appear as a paradox, since the same product 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, there is a limit on the number of parallel activities that can 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 for some time since it would not be possible to give them any work. These idle times would show up as higher effort and larger cost.

From Figure 3.5, we can observe that 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, semidetached, or embedded type. (Please verify this using the basic COCOMO formulas provided above) We can interpret it to mean that there is more scope for parallel activities for system and utility programs than those in application programs.

From the effort estimation, the project cost can be obtained by multiplying the required effort by the manpower cost per month. But, implicit in this project cost computation is the assumption that the entire project cost is incurred on account of the manpower cost alone. In addition to manpower cost, a project would incur costs due to hardware and software required for the project and the company overheads for administration, office space, etc.

Another point should be carefully remembered. From the effort estimation for a project expressed in programmer-months and the nominal development time, the novices would de-

termine the staffing level by a simple division. However, we are going to examine the staffing problem in more detail in Section 3.8. From the discussion in Section 3.8 it would become clear that the simple division approach to obtain the staff size is highly improper.

Example 3.1 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 developers 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

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

3.6.2 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 required to develop the product as well as the development time. 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 recognizes this fact and refines the initial estimate obtained using the basic COCOMO expressions by using a set of 15 cost drivers (multipliers) based on various attributes of software development. For example, if modern programming practices are used, the initial estimates are scaled downward by multiplication with a cost driver having a value less than 1. If there are stringent reliability requirements on the software product, this initial estimate is scaled upward. Boehm requires the project manager to rate these 15 different parameters for a particular project on a scale of one to three. Then, depending on these ratings, he suggests appropriate cost driver values which should be multiplied with the initial estimate obtained using the basic COCOMO. In general, the cost drivers can be classified as being attributes of the following items:

1. Product: The characteristics of the product that are considered include the inherent complexity of the product, reliability requirements of the product, etc.

2. Computer: Characteristics of the computer that are considered include the execution speed required, storage space required, etc.

3. Personnel: The attributes of development personnel that are considered include the experience level of personnel, programming capability, analysis capability, etc.

4. Development environment: Development environment attributes capture the development facilities available to the developers. An important parameter that is considered is the sophistication of the automation (CASE) tools used for software development.

We have discussed only the basic ideas behind the COCOMO model. A detailed discussion on the COCOMO model are beyond the scope of this book and the interested reader may refer [Boehm, 81].

3.6.3 Complete COCOMO

A major shortcoming of both the basic and the intermediate COCOMO models is that they consider a software product as a single homogeneous entity. However, most large systems are made up of several smaller subsystems. These subsystems may have widely different characteristics. For example, some subsystems may be considered as organic type, some semidetached, and some embedded. Not only that the inherent development complexity of the subsystems may be different, but also for some subsystem the reliability requirements may be high, for some the development team might have no previous experience of similar development, and so on. The complete COCOMO model considers these differences in characteristics of the subsystems and estimates the effort and development time as the sum of the estimates for the individual subsystems. The cost of each subsystem is estimated separately. This approach reduces the margin of error in the final estimate.

Let us consider the following development project as an example application of the complete COCOMO model. A distributed Management Information System (MIS) product for an organization having offices at several places across the country can have the following subcomponents:

- Database part
- Graphical User Interface (GUI) part
- Communication part

Of these, the communication part can be considered as embedded software. The database part could be semidetached software, and the GUI part organic software. The costs for these three components can be estimated separately, and summed up to give the overall cost of the system.

To improve the accuracy of their results, the different parameter values of the model can be fine-tuned and validated against an organization's historical project database to obtain more accurate estimations. Estimation models such as COCOMO are not accurate and lack a full scientific justification. Still, software cost estimation models such as COCOMO are required for an engineering approach to software project management. Companies consider computed estimates to be satisfactory, if these are within about 80% of final cost. Although these estimates are gross approximations—without such models, one has only subjective judgements to rely on.

3.6.4 COCOMO 2

Since the time that COCOMO estimation model proposed in the early 1980s, both the software development paradigm and problems have undergone a sea change. The present day software projects are much larger in size and reuse of existing software to develop new products has become pervasive. This has given rise to component-based development. New life cycle models and development paradigms are being deployed for web-based and component-based software. During the 1980s rarely any program was interactive, and graphical user interfaces were almost non-existent. On the other hand, most of the present day software is highly interactive and has elaborate graphical user interface. To make COCOMO suitable in the changed scenario Boehm proposed COCOMO 2 [Boehm, 95].

COCOMO 2 provides three increasingly detailed cost estimation models. These can be used to estimate project costs at different phases of the software. As the project progresses through these models can be applied at the different stages of the same project.

1. Application composition: This model as the name suggests, can be used to estimate cost for prototyping, e.g. to resolve user interface issues.

2. Early design: This supports estimation of cost at the architectural design stage.

3. Post-architecture stage: This provides cost estimation during detailed design and coding stage.

The post-architectural model can be considered as an update of the original COCOMO.