4.4 THE CONSTRUCTIVE COST MODEL (COCOMO)

This model gained rapid popularity following the publication of B.W. Boehm's excellent book *Software Engineering Economics* in 1981 [BOEH81]. COCOMO is a hierarchy of software cost estimation models, which include basic, intermediate and detailed sub models.

4.4.1 Basic Model

The basic model aims at estimating, in a quick and rough fashion, most of the small to medium sized software projects. Three modes of software development are considered in this model: organic, semi-detached and embedded.

In the organic mode, a small team of experienced developers develops software in a very familiar environment. The size of the software development in this mode ranges from small

(a few KLOC) to medium (a few tens of KLOC), while in other two modes the size ranges from small to very large (a few hundreds of KLOC).

In the embedded mode of software development, the project has tight constraints, which might be related to the target processor and its interface with the associated hardware. The problem to be solved is unique and so it is often hard to find experienced persons, as the same does not usually exist.

The *semi detached* mode is an intermediate mode between the organic mode and embedded mode. The comparision of all three modes is given in Table 4.4.

Table 4.4: The comparison of three COCOMO modes

Mode	*Project size*	*Nature of project*	*Innovation*	*Deadline of the project*	*Development environment*
Organic	Typically 2 – 50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & In house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar projects. For Example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For Example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer Interfaces required

Depending on the problem at hand, the team might include a mixture of experienced and less experienced people with only a recent history of working together. The basic COCOMO equations take the form

$$E = a_b(KLOC)^{b_b} \quad (4.8)$$

$$D = c_b(E)^{d_b} \quad (4.9)$$

where E is effort applied in Person-Months, and D is the development time in months. The coefficients a_b, b_b, c_b and d_b are given in Table 4.4(*a*).

Table 4.4(a): Basic COCOMO co-efficients

Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

When effort and development time are known, the average staff size to complete the project may be calculated as:

$$\text{Average staff size (SS)} = \frac{E}{D} \text{ Persons.}$$

When project size is known, the productivity level may be calculated as:

$$\text{Productivity (P)} = \frac{KLOC}{E} \text{ KLOC/PM.}$$

With the basic model, the software estimator has a useful tool for estimating quickly, by two runs on a pocket calculator, the cost and development time of a software project, once the size is estimated. The software estimator will have to assess by himself/herself which mode is the most appropriate.

Example 4.5

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

The basic COCOMO equations take the form:

$$E = a_b(KLOC)^{b_b}$$

$$D = c_b(KLOC)^{d_b}$$

Estimated size of the project = 400 KLOC

(*i*) **Organic mode**

$$E = 2.4(400)^{1.05} = 1295.31 \text{ PM}$$

$$D = 2.5(1295.31)^{0.38} = 38.07 \text{ M}$$

(*ii*) **Semidetached mode**

$$E = 3.0(400)^{1.12} = 2462.79 \text{ PM}$$

$$D = 2.5(2462.79)^{.35} = 38.45 \text{ M}$$

(*iii*) **Embedded mode**

$$E = 3.6(400)^{1.20} = 4772.81 \text{ PM}$$

$$D = 2.5(4772.8)^{.32} = 38 \text{ M}$$

As we have seen, effort calculated for embedded mode is approximately 4 times, the effort for organic mode. However, the effort calculated for semidetached mode is 2 times the effort of organic mode. There is a large difference in these values. But, surprisingly, the development time is approximately the same for all three modes. It is clear from here that the

selection of mode is very important. Since, development time is approximately the same, the only varying parameter is the requirement of persons. Every mode will have different manpower requirement. If we look to the Table 4.4, it is mentioned that for over 300 KLOC projects, embedded mode is the right choice. The selection of a mode is not only dependent on project size, but also on other parameters as mentioned in Table 4.4. We should be utmost careful about the selection of mode for the project.

Example 4.6

A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the effort, development time, average staff size and productivity of the project.

Solution

The semi-detached mode is the most appropriate mode; keeping in view the size, schedules and experience of the development team.

Hence $E = 3.0(200)^{1.12} = 1133.12$ PM

$D = 2.5(1133.12)^{.35} = 29.3$ M

$$\text{Average staff size (SS)} = \frac{E}{D} \text{ Persons}$$

$$= \frac{1133.12}{29.3} = 38.67 \text{ Persons.}$$

$$\text{Productivity} = \frac{KLOC}{E} = \frac{200}{1133.12} = 0.1765 \text{ KLOC/PM}$$

$P = 176$ LOC/PM.

4.4.3 Detailed COCOMO Model

A large amount of work has been done by Boehm to capture all significant aspects of a software development. It offers a means for processing all the project characteristics to construct a software estimate. The detailed model introduces two more capabilities:

1. Phase-sensitive effort multipliers:

   Some phases (design, programming, integration/test) are more affected than others by factors defined by the cost drivers. The detailed model provides a set of phase sensitive effort multipliers for each cost driver. This helps in determining the manpower allocation for each phase of the project.

2. Three-level product hierarchy:

   Three product levels are defined. These are module, subsystem and system levels. The ratings of the cost drivers are done at appropriate level; that is, the level at which it is most susceptible to variation.

Development phases

A software development is carried out in four successive phases: plans/requirements, product design, programming and integration/test.

1. **Plan/requirements:** This is the first phase of the development cycle. The requirement is analyzed, the product plan is set up and a full product specification is generated. This phase consumes from 6% to 8% of the effort and 10% to 40% of the development time. These percentages depend not only on mode (organic, semi-detached or embedded), but also on the size.
2. **Product design:** The second phase of the COCOMO development cycle is concerned with the determination of the product architecture and the specification of the subsystem. This phase requires from 16% to 18% the nominal effort and can last from 19% to 38% of the development time.

3. **Programming:** The third phase of the COCOMO development cycle is divided into two sub phases: detailed design and code/unit test. This phase requires from 48% to 68% of the effort and lasts from 24% to 64% of the development time.
4. **Integration/Test:** This phase of the COCOMO development cycle occurs before delivery. This mainly consists of putting the tested parts together and then testing the final product. This phase requires from 16% to 34% of the nominal effort and can last from 18% to 34% of the development time.

4.5 COCOMO-II

COCOMO-II is the revised version of the original COCOMO (discussed in article 4.4) and is developed at University of Southern California under the leadership of Dr. Barry Boehm. The model is tuned to the life cycle practices of the 21st century. It also provides a quantitative analytic framework, and set of tools and techniques for evaluating the effects of software technology improvements on software life cycle costs and schedules. The following categories of applications/projects are identified by COCOMO-II for the estimation [UCSD01] and are shown in Fig. 4.4.

(*i*) **End user programming:** This category is applicable to small systems, developed by end user using application generators. Some application generators are spreadsheets, extended query system, report generators etc. End user may write small programs using application generators. The end users may not have sufficient knowledge about computers and software engineering practices. However, they may have in-depth knowledge about their business needs and practices. Hence, this excellent domain knowledge may motivate them to develop an application using user friendly tools like MS-Excel, MS-Access, MS-Studio etc.

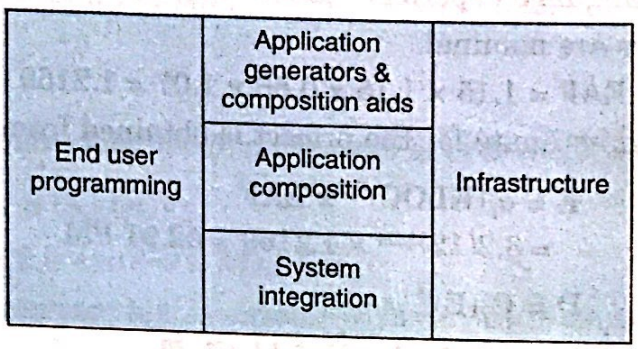

Fig. 4.4: Categories of applications/projects

(*ii*) **Infrastructure sector:** This category is applicable to the infrastructure development *i.e.*, the software that provides infrastructure like operating systems, database management systems, user interface management system, networking system etc. Some commercial examples are Microsoft products, Oracle, DB2, MAYA, LINUX, 3D-STUDIO.

Infrastructure developers generally have good knowledge of software development and software engineering practices and relatively little knowledge about applications. The product lines will have many reusable components, but the pace of technology (new processor, memory, communications, display, and multimedia technology) will require them to build many components and capabilities from scratch.

(*iii*) **Intermediate sectors:** This category is partitioned in three sub categories as shown in Fig. 4.4. Software developers will need to have good knowledge of software development and software engineering practices, experience and expertise of infrastructure software and indepth domain knowledge of one or more applications. Creating this talent pool is a major challenge.

Application generators and composition aids : This subcategory will create largely prepackaged capabilities for user programming. Typical firms operating in this sector are Microsoft, Lotus, Novell, Borland, Alias Wavefront, Oracle, IBM. Their product lines will have many reusable components, but also will require a good deal of new capability development from the scratch. Application composition aids will be developed both by the firms above and by software product line investments of firms in the application composition sector.

Application composition sector : This subcategory deals with applications which are too diversified to be handled by prepackaged solutions, but which are sufficiently simple to be rapidly composable from interoperable components. Typical components will be graphic user interface (GUI) builders, databases or object managers, domain specific components such as financial, medical, or industrial process control packages etc.

These applications are complex, large, versatile, diversified and require specialised developers with sound knowledge of development and software engineering practices.

However, they are developed using application generator environment like CASE tools, DBMS (DB2, oracle etc.), and 4GL programming tools (Developer 2000, Visual basic, Power builder, ASP, JSP, PHP etc).

System integration : This subcategory deals with large scale, highly embedded, or unprecedented systems. Portions of these systems can be developed with application composition capabilities, but their demands generally require a significant amount of upfront systems engineering and customised software development activities.

Stages of COCOMO-II : The end user programming sector does not need a COCOMO-II model. Its applications are normally developed in hours to days. Hence a simple activity based estimate will generally be sufficient.

COCOMO-II includes three stages. Stage I supports estimation of prototyping or application composition types of projects. Stage II supports estimation in the early design stage of a project, when less is known about the project's cost drivers. Stage III supports estimation in the Post-Architecture stage of a project. The details are given in Table 4.8.

Table 4.8: Stages of COCOMO-II

Stage No.	*Model name*	*Applicable for the types of projects*	*Applications*
Stage I	Application composition estimation model	Application composition	In addition to application composition type of projects, this model is also used for prototyping (if any) stage of application generators, infrastructure & system integration.
Stage II	Early design estimation model	Application generators, infrastructure & system integration.	Used in early design stage of a project, when less is known about the project.
State III	Post architecture estimation model	Application generators, infrastructure & system integration	Used after the completion of the detailed architecture of the project