CHAPTER 4

EFFORT ESTIMATION ANALYSIS

4.1 INTRODUCTION

Corporate are moving forward from an industrial based society to a knowledge based society. They are facing a major issue on how to use the training systems to establish a continuous learning philosophy in training. Corporate view continuous learning as the key to the competitive merits and training is seen as one of the elements of the huge orientation across continuous improvement. The emerging knowledge and skills demand geographic spreading of workspace requiring minimal costs, accessibility for anytime/anywhere way of learning. Here, e-learning provides solution to this scenario to learn anywhere/anytime. However many educational institutions and organization have not updated the infrastructure and resources to run e-learning solutions.

Many organizations are working to expand e-learning interventions as it becomes more complex to assess their readiness to enhance technology for successful implementation the system successfully. The learning strategies are adjusted with the help of e-learning effort estimation. Also, the previous failures of e-learning interferences show the way to implement a comprehensive readiness assessment to minimize the risk. Still, none of the effort estimation techniques are accepted by the community since none of them delivers reliable for diverse environments. Each technique is suitable for some specific domain due to their nature and characteristics.

This chapter proposes an effort estimation technique which is suitable for e-learning effort estimation. The proposed model is formulated
with the help of Functional Point Analysis (FPA) with the basic COCOMO model. Finally, the Linear Regression Method is applied across the estimation of effort to show the relationship between the function point and effort estimates.

4.2 CONCEPT OF METHODOLOGY

Effort estimation is the major part in the e-learning project development process. The software sizing techniques are classified into code based sizing and functional size measurements. The code based sizing metric estimates the size and complexity of the project based on the programmed source code. Due to the dedication of essential amount of effort to programming, suitable measure correctly quantifying the code is accepted as perceivable indicator of the software cost. The baseline size of the effort estimation model is the count of new Line of Code (LOC). SLOC is a code based sizing metric. SLOC is referred to as Source LOC that is delivered as the portion of the project which is included as drivers and other necessary software. Code size is defined in terms of thousands of SLOC (KLOC). The objective is to evaluate the amount of intellectual work considered in the project development. Evaluation of SLOC takes the new LOC into account. There are two possibilities:

1. To count the SLOC checklists or supported tools (or)
2. To count the Unadjusted Function Points (UFPs) to translate them through backfiring table to SLOC.

SLOC is largely accepted for the following reasons:

- It is highly concurrent with the software cost.
- They are appropriate inputs for software estimation models.
• It can be precisely and easily counted with the help of software tools which eliminate inconsistencies in SLOC counts.

The basic COCOMO model suites permit a powerful weapon to predict the software costs. Reliable size estimation is very much essential for a good model estimate. Size estimation is still a challenging task due to the projects that are usually developed with the new, re-used and automatically converted codes. FPA method is an alternative approach to code based sizing methods. It considers the static and dynamic features of the sizing system. The static feature is denoted by the data accessed or stored by the system. The dynamic feature reproduces the transactions performed to manipulate and access the data.

4.3 ESTIMATION ANALYSIS ON E-LEARNING PROJECT DEVELOPMENT TOOLS

From the relevant literature, COCOMO is identified as suitable for effort estimation of e-learning projects (Moseley and Valverde 2014). COCOMO appear in three levels such as,

• Basic COCOMO,

• Intermediate COCOMO and

• Advanced or detailed COCOMO

The basic COCOMO model defines a single valued, static model which estimates the project development time and cost as functions of the essential program size expressed in terms of thousands of delivered source instructions (KDSI). The intermediate COCOMO model calculates the effort as the function of the program size and the group of 15 cost drivers which consist of the subjective assessments of hardware personnel, project attributes
and the product. An *advanced or detailed COCOMO model* combines all the properties of the intermediate version with the cost driver’s impact on each stage such as design, analysis, etc. of the software engineering process.

Three types of COCOMO models are applied in the proposed effort estimation model with each of the levels progressively providing precise estimates. Boehm proposed that any software development project can be categorized into any of the following three classes which is classified based on their complexity level:

1. **Organic**
2. **Semi-detached** and
3. **Embedded.**

Boehm not only takes the characteristics of the product, but also considers the development team and environment. These three classes are based on the application, system programs and utility.

**Organic** – A project can be classified as organic, if it means formulating with an easy and well understood application program, the size of the team is reasonably small and the developing members are well versed in designing similar types of projects.

**Semi-detached** – A project can be classified as semi-detached, if it consists of the combination of experienced and inexperienced developers. Developers have only limited experience on the related systems, not familiar with some aspects of the project being created.
**Embedded** – A project can be classified as embedded, if it is strongly coupled with the complex hardware or rigorous regulations on the operational strategies.

In this research work, the e-learning projects are categorized into simple, medium, and complex projects based on their content. Details are provided in Chapter 3. The text-based e-learning courses are categorized into simple projects and the contents are text-based with hyperlinks and other materials. Figure 4.1 shows an easy example of simple e-learning project. It uses only the text-based contents with some illustrative diagrams.

![Figure 4.1 Example of simple e-learning project (text + image based)](image)

Animation-based e-learning courses are categorized into medium projects and the contents are moving illustrations or diagrams coupled with audio. Figure 4.2 shows an example of medium e-learning project. It uses the whiteboard animation to showcase the author's resume and his career background.
Figure 4.2 Example of medium e learning project (text + audio based)

Video based courses and the combination of text, audio and video based courses are categorized into complex projects.

Figure 4.3 Example for complex e learning project (text + audio + video based)
Usually, the contents are full motion video clippings of lecture or a dramatized sequence. Figure 4.3 shows an example of a complex project. The video based online course contains the text, audio and video contents for e-learners.

Organic mode calculation is applied for small projects, the semi-detached calculation is applied for medium projects and the embedded calculation is applied for complex projects. Table 4.1 shows the development modes of the basic COCOMO model. It clearly illustrates the size, innovation, constraints and deviation environment for the three operating modes.

**Table 4.1 Development Modes of Basic COCOMO Model**

<table>
<thead>
<tr>
<th>S. No</th>
<th>Development Mode</th>
<th>Size</th>
<th>Innovation</th>
<th>Constraints</th>
<th>Dev. Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Organic</td>
<td>Small</td>
<td>Little</td>
<td>Not tight</td>
<td>Stable</td>
</tr>
<tr>
<td>2.</td>
<td>Semi-detached</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>3.</td>
<td>Embedded</td>
<td>Large</td>
<td>Greater</td>
<td>Tight</td>
<td>Complex hardware/customer interfaces</td>
</tr>
</tbody>
</table>

The basic COCOMO model gives a fairly accurate estimate of the e-learning project parameters. It is expressed based on the following equation:

\[
\text{Effort} = k_1 \times (KLOC)^{k_2}PM
\]  

(4.1)

\[
\text{Tdev} = l_1 \times (\text{Effort})^{l_2} \text{ Months}
\]  

(4.2)

Here,

- KLOC denotes the estimated size of the e-learning project and it is expressed in terms of kilo lines of code.
• $k_1$, $k_2$, $l_1$, $l_2$ denotes the constants for each category of e-learning projects.

• Tdev denotes the estimated duration for designing the project and it is expressed in terms of months.

• Effort denotes the total duration amount of effort required designing the project and it is expressed in terms of Person Months (PMs).

4.4 PROCEDURE FOR CALCULATING FUNCTION POINT ANALYSIS

Function Point Analysis (FPA) is the standard metric for measuring the functional size of a software system. The first work to be published about function point was written in the late 1970s by A.J. Albrecht of IBM, for a transaction-oriented system (Futrell, et al. 2001). FPA is used to predict the effort estimation of the software project in the early stage of the life cycle. It measures the complexity of the functions and overcome the difficulties of Line of Code. FPA helps the developers and users to quantify the size and complexity of software application functions in a way that is useful to software users (Furey 1997).

Objectives of FPA

FPA measures software by calculating the functionality the software delivers to the user based principally on logical intent. The main objectives of FPA are listed as follows:

• It estimates the software growth and maintains autonomy of technology applied for execution.

• It estimates the functionality that the user requires and receives.
- A consistent measure between the different organizations and projects.
- It is easy to reduce the overhead of the measurement procedure.

4.4.1 Procedure Diagram for Function Point Counting

![Diagram]

**Figure 4.4 Procedure Diagram for function point analysis**

Figure 4.4 shows the procedure diagram for Function Point Counting (FPC).

**Step 1: Determine type of count**

The initial step is to calculate the type of function point count. It can be associated with either applications or projects. There are three kinds of FPCs:
1. Development project FPC
2. Enhancement project FPC
3. Application FPC

**Step 2: Identify the counting scope and boundary**

The counting point describes the functionality that is added in the particular FPC. The application boundary denotes the limit between the software being estimated and the user.

**Step 3: Determine the Unadjusted FPC (UFPC)**

Unadjusted function point specifies the total number of function points depending on the following two factors:

1. Data functions
2. Transaction functions.

![Diagram of UFPC](image)

**Figure 4.5 Factors of UFPC**

Figure 4.5 illustrates the two factors and their types of UFPC.

**Step 3.1: Count Data Functions**

There are two types of data functions. They are:
1. Internal logical file (ILF): ILF is a user identifiable group of logically related data or control information maintained within the boundary of the application.

2. External Interface File (EIF): EIF is a user identifiable group of logically related data or control information referred by the application, but maintained within the boundary of another application (SE 2010).

**Step 3.2: Count Transaction Functions**

There are three types of transaction functions. They are:

1. External Input (EI): External Inputs are received from the user to the software which provides the application-oriented data.

2. External Output (EO): Things are provided by the software that goes to the outside the system like screen data, report data, error message and so on.

3. External inquires (EI): Inquiries may be the command or requests generated from outside and involving direct access to a database that retrieve the information (SE 2010).

Table 4.2 shows the computing procedure for UFP for the five categories of data and transaction functions. Table 4.2 represent a constant value which is assigned to each other depending on the complexity of the code.
Table 4.2 UFP Calculations

<table>
<thead>
<tr>
<th>Sno</th>
<th>Function Type</th>
<th>Weight by Functional Complexity</th>
<th>Total (FP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>External Input (EI’s)</td>
<td>Low __ x 3</td>
<td>Total = count of EI x 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average __ x 4</td>
<td>Total = count of EI x 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High __ x 6</td>
<td>Total = count of EI x 6</td>
</tr>
<tr>
<td>2.</td>
<td>External Output (EO’s)</td>
<td>Low __ x 4</td>
<td>Total = count of EO x 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average __ x 5</td>
<td>Total = count of EO x 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High __ x 7</td>
<td>Total = count of EO x 7</td>
</tr>
<tr>
<td>3.</td>
<td>External Enquiry (EI’s)</td>
<td>Low __ x 3</td>
<td>Total = count of EI x 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average __ x 4</td>
<td>Total = count of EI x 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High __ x 6</td>
<td>Total = count of EI x 6</td>
</tr>
<tr>
<td>4.</td>
<td>Internal Logical File (ILF)</td>
<td>Low __ x 7</td>
<td>Total = count of ILF x 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average __ x 10</td>
<td>Total = count of ILF x 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High __ x 15</td>
<td>Total = count of ILF x 15</td>
</tr>
<tr>
<td>5.</td>
<td>External Interface File (EIF)</td>
<td>Low __ x 5</td>
<td>Total = count of EIF x 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Average __ x 7</td>
<td>Total = count of EIF x 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High __ x 10</td>
<td>Total = count of EIF x 10</td>
</tr>
</tbody>
</table>

Total Number of UFPs: Sum of the total FP

Step 4: Determine the Value Adjusted Factors (VAF)

VAF represents the general functionality given to the user of the corresponding application. It consists of 14 General System Characteristics (GSCs). Each characteristic has related definitions that assist the degree of influence of the characteristic. It ranges between the scales of 0 to 5 categorizing from no influence to strong influence.

General System Characteristics (GSCs)

The next step following calculation of the unadjusted function point, involves gathering information about the environment and complexity of the project or application and scale from 0 to 5 (degree of influence) (Vickers and Street 2001).
Degree of Influence

Each characteristic should be examined by its Degree of Influence (DI) based on the scale from 0 to 5. Table 4.3 shows the scale value and its influence. The 14 characteristics are illustrated Table 4.4.

**Table 4.3 Degree of Influence**

<table>
<thead>
<tr>
<th>SNo.</th>
<th>Scale</th>
<th>Influence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0</td>
<td>No influence</td>
</tr>
<tr>
<td>2.</td>
<td>1</td>
<td>Incidental influence</td>
</tr>
<tr>
<td>3.</td>
<td>2</td>
<td>Moderate influence</td>
</tr>
<tr>
<td>4.</td>
<td>3</td>
<td>Average influence</td>
</tr>
<tr>
<td>5.</td>
<td>4</td>
<td>Significant influence</td>
</tr>
<tr>
<td>6.</td>
<td>5</td>
<td>Strong influence</td>
</tr>
</tbody>
</table>

**Table 4.4 General System Characteristics (Vickers and Street 2001)**

<table>
<thead>
<tr>
<th>Sno.</th>
<th>General System Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Data Communication</td>
</tr>
<tr>
<td>2.</td>
<td>Distributed Data Processing</td>
</tr>
<tr>
<td>3.</td>
<td>Is performance critical?</td>
</tr>
<tr>
<td>4.</td>
<td>Heavily Used configuration</td>
</tr>
<tr>
<td>5.</td>
<td>Transaction Rate</td>
</tr>
<tr>
<td>6.</td>
<td>Online Data Entry</td>
</tr>
<tr>
<td>7.</td>
<td>End-User Efficiency</td>
</tr>
<tr>
<td>8.</td>
<td>Online Update</td>
</tr>
<tr>
<td>9.</td>
<td>Complex Processing</td>
</tr>
<tr>
<td>10.</td>
<td>Reusability</td>
</tr>
<tr>
<td>11.</td>
<td>Installation Ease</td>
</tr>
<tr>
<td>12.</td>
<td>Operational Ease</td>
</tr>
<tr>
<td>13.</td>
<td>Multiple Site</td>
</tr>
<tr>
<td>14.</td>
<td>Facility Change</td>
</tr>
</tbody>
</table>
Data Communication

    Data communication defines the degree of direct interaction of the system with other applications. The data and control details are sent and received through suitable communication services. Terminals associated with the control unit are considered in the utilization of the communication services. All the data communication connection needs some kind of protocol, which is a set of gathered information that allows data transfer between two devices or systems. Table 4.4 describes the score definition for data communication.

Distributed Data Processing

    Distributed data processing defines the degree of forwarding of application data or sharing of the data among the components of the application. Distributed data/processing functions are the feature of the application inside the application boundary.

Performance

    It defines the degree of influence seeming the response time and throughput performance by the development of the application. The performance objectives are approved by the user in throughput or response, influence the development, design, support and the installation of the application.

Heavily Used Configuration

    It defines the degree of influence seeming the computer resource limitations on the application development. A largely applied operational configuration needs special design considerations. The user needs to execute the application on committed device which will greatly be used.
Transaction Rate

It defines the degree of influence seeming the rate of business transactions the application development. The rate is high and it influences the development, design, support and installation of the application.

Online Data Entry

It defines the degree of entry of the data with the help of interactive interactions. An online data entry and control functions are given in the application.

End-User Efficiency

It defines the degree of deliberation for the ease of use and human factors for the user of the estimated application. The online functions highlight the design for end user efficiency. The end user design includes the following things:

- Menus
- Navigational supports such as function keys, jumps and dynamically generated menus
- Scrolling
- Pre-assigned function keys
- Online help and documents
- Automated cursor movement
- Cursor selection of screen data
- Remote printing through online transactions
• Pop-up windows
• Heavy use of highlighting, color underlining, reverse video and other indicators
• Bilingual and multilingual supports
• As few screens as possible to establish a business function

Online Update

It defines the degree of updating seen in the ILFs online. So, the application offers online update for the ILFs.

Complex Processing

It defines the degree of influence seeming which the processing logic influenced the application development. The set of following components are present:

• Sensitive control/ application specific security processing
• Extensive mathematical processing
• Extensive logical processing
• Incomplete transaction due to the exception processing that should be processed again.
• Complex processing to manage the multiple I/O possibilities.

Reusability

It defines the degree of specialty in design of the code and the application formulation and support which can be used in other applications.
Installation Ease

It defines the degree of influence seeming conversion from the previous settings the application development. Installation and conversion ease are the application characteristics. The plan and conversion tools are given and tested at the time of system test phase.

Operational Ease

It defines the degree of influence seeming the application attends to operational concern like start up, recovery and backup processes. The application reduces the requirement of the manual actions such as paper handling, tape mounts and direct on location manual interference.

Multiple Sites

It defines the degree of influence seeming the application has been designed for multiple locations and user associations. It has been successfully formulated, developed and managed to be installed at various sites for various organizations.

Facilitate Change

- Flexible query and report ability are given to help management of the simple requests such as logic applied to only one ILF.
- Flexible query and report ability are given to help management of requests of average complexity such as logic applied to more than one ILF.
- Flexible query and report ability are given to help management of complex requests such as logic mixture on one or more ILF.
• Business control data is reserved in tables which are managed by the user with the online interactive processes. However, modifications take effect only on the following business day.

• Business control data is reserved in tables which are managed by the user with the online interactive processes and the modifications take effect immediately.

Subsequent to all the 14 GSC’s, the Value Adjustment Factors (VAF) is calculated. The formulae that is used to calculate the VAF using equation (4.3) is

\[
VAF = 0.65 + (0.01 \times \sum_{i=0}^{n} C_i)
\]  

(4.3)

**Step 5: Calculate the Adjusted FPC (AFPC)**

After calculating the UFP with transactional and data function types and obtaining the VAF values, finally the Function Point Count (FP) is calculated. The formulae for calculating the Function Point is shown in the equation (4.4).

\[
FP = UFP \times VAF
\]  

(4.4)

The lines of code can be calculated for the following equation (4.5).

\[
KLOC = 37.65 \times FP
\]  

(4.5)

The constant value of 37.65 is average value of the LOC per function point derived from the standard table of conversion from the programming language to SLOC per function point for multi-language support.
4.5 EFFORT ESTIMATION OF E-LEARNING PROJECTS USING BASIC COCOMO

After obtaining the results of the functional size of the e-learning projects, it is necessary to calculate the effort, time and average staff required to complete the project using basic COCOMO model. In the basic COCOMO model the effort, time and cost of the e-learning projects by using only lines of code.

Effort estimation is calculated for simple projects using Boehm’s COCOMO model for organic projects and is depicted in equation (4.6).

\[
\text{Effort} = 2.4 \times (\text{KLOC})^{1.05} \text{Per Month} \quad (4.6)
\]

Effort estimation is calculated for medium projects using Boehm’s COCOMO model for semi-detached projects and is depicted in equation (4.7).

\[
\text{Effort} = 3.0 \times (\text{KLOC})^{1.12} \text{Per Month} \quad (4.7)
\]

Effort estimation is calculated for complex projects using Boehm’s COCOMO model for embedded projects and is depicted in equation (4.8).

\[
\text{Effort} = 3.6 \times (\text{KLOC})^{1.20} \text{Per Month} \quad (4.8)
\]

The estimation of development time can be calculated using organic, semi-detached and embedded formulae which is used for the all the three categories of the projects.

The effort estimation of development for simple projects can be considered by organic types, as only the e-learning text content is involved in these projects. The development team can be of a small size because it is easy to create the content by subject experts and it can be easily implemented by
instructional designer. The development times needed for the simple projects are calculated the formulae given in the equation (4.9).

\[
\text{Organic} \quad : \quad T_{\text{dev}} = 2.5(\text{Effort})^{0.38} \quad (4.9)
\]

Effort estimation of the medium projects can be considered by Semi-detached types, as it involves text, animation, audio and video etc. It can be a mixture of the developers like content developer, animation and give the effects to the pictures etc. The development times needed for the medium projects are calculated using formulae given in the equation (4.10).

\[
\text{Semi-detached} : T_{\text{dev}} = 2.5(\text{Effort})^{0.35} \quad (4.10)
\]

Complex projects are like creation of complex animations, simulation and action scripts etc., are considered by embedded types. The development times needed for the projects are calculated using the formulae given in the equation (4.11).

\[
\text{Embedded} \quad : \quad T_{\text{dev}} = 2.5(\text{Effort})^{0.32} \quad (4.11)
\]

In the three types of projects, the estimation of development time is calculated according to Boehm definition of organic, semi-detached and embedded.

### 4.6 INTERMEDIATE COCOMO MODEL IN E-LEARNING PROJECTS

Intermediate COCOMO model utilizes the size and modes as similar to the basic COCOMO model. Additionally, it has 15 variables called cost drivers and the modified effort equations. KLOC and cost driver ratings are given as the input to this model which improves the effort estimates. The intermediate COCOMO formula is given equation (4.12).
\[ Effort (E) = a \times (size)^b \times C \]  
(4.12)

In this equation, the constants for coefficients and exponents vary for each mode (a and b are coefficients). This model also has three types of modes as similar to the basic model with some modifications. The equations are:

Effort calculation for organic mode,
\[ E = 3.2 \times (size)^{1.05} \times C \]  
(4.13)

Effort calculation for semidetached mode,
\[ E = 3.0 \times (size)^{1.12} \times C \]  
(4.14)

Effort calculation for embedded mode,
\[ E = 2.8 \times (size)^{1.20} \times C \]  
(4.15)

### 4.6.1 Cost Drivers

An Effort Adjustment Factor (EAF) refers the effect of increasing/decreasing the effort and hence the cost is based on the set of environmental factors. These factors are also called as cost drivers or cost adjustment factors (\(C_i\)s)

**Steps to determine the multiplying factor:**

1. Assign the numerical factors to the cost drivers.
2. Multiply the cost factors collectively to formulate the EAF.

Collective multiplication of the cost factors, it affects the project schedule and cost estimation by 10 times or even more. Product of cost drivers is given as follows,

\[ EAF = C_1 \times C_2 \times \ldots \times C_n \]  
(4.16)
$C_i = i^{th}$ cost adjustment factor

$C_i = 1$ represents the cost driver does not apply

$C_i > 1$ represents increased cost due to this factor

$C_i < 1$ represents decreased cost due to this factor.

### 4.6.2 Categories of Cost Drivers

Cost drivers are classified into four categories. They are:

1. Product Attributes
2. Computer Attributes
3. Project Attributes
4. Personnel Attributes

**Product Attributes** – Some of the attributes are used to enhance or laying down the project cost depending on the nature of the project to be done or the project itself. Those attributes include:

- Essential reliability – mainly related to real time applications
- Product complexity – mainly related to execution time constraints
- Database size – mainly related to data processing applications

**Computer Attributes** – These attributes use the computer platform as the supporting tool to do with the project to be done.

- Execution time constraints – seen when the processor speed is hardly sufficient
- Virtual machine volatility – seen to the hardware and operating system of the target system
- Computer turnaround time – applied for development
- Main storage constraints – seen when the memory size is hardly sufficient

**Project Attributes** – These attributes are related to tools and practices.
- Modern programming practices – Uses structures method or object oriented ones.
- Schedule compression / expansion – divergence from ideal can never assist, but shorter is worse than longer.
- Modern programming tools – uses good debuggers, CASE, test generation tools.

**Personnel Attributes** – These attributes describe the person who do the job. These factors have the tendency to increase or decrease the cost.
- Programmer capability
- Analyst capability
- Application experience
- Programming language experience – uses tools and practices
- Virtual machine experience – uses hardware and operating system.

**Other Cost Drivers** – Some of the additional attributes are sometimes added by the project manager of other company strengths or weaknesses. These include:
- Development of machine volatility – compilers, unstable OS, CASE tools etc.
• Requirement volatility – few is expected, but too much can be a great issue

• Access to data - sometimes, it is very difficult

• Security requirements – applied for classified programs

• Impact of physical surroundings

• Impact of imposed methods and standards

**Table 4.5 Categories of Intermediate COCOMO Cost Driver**

<table>
<thead>
<tr>
<th>Product</th>
<th>Computer</th>
<th>Personnel</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required Software Reliability</td>
<td>Execution Time Constraint (TIME)</td>
<td>Analyst Capability (ACAP)</td>
<td>Use of Modern Programming Practices (MODP)</td>
</tr>
<tr>
<td>(RELY)</td>
<td>Main Storage Constraint (STOR)</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>Database Size (DATA)</td>
<td>Virtual Machine Volatility (VIRT)</td>
<td>Experience (AEXP) Programmer</td>
<td>Use of Software Tools (TOOL)</td>
</tr>
<tr>
<td>Product Complexity (CPLX)</td>
<td>Computer Turnaround Time (TURN)</td>
<td>Capability (PCAP) Virtual Machine</td>
<td>Required Development Schedule (SCED)</td>
</tr>
</tbody>
</table>

Table 4.5 illustrates the set of the attributes with their corresponding categories. The product of cost drivers are given as follows:

\[
C = RELY \times DATA \times CPLX \times TIME \times STOR \times VIRT \times \\
TURN \times ACAP \times AEXP \times PCAP \times VEXP \times LEXP \times MODP \times \\
TOOL \times SCED
\]  

(4.17)
Each cost driver estimates a multiplying factor which determines the effect of the attribute on the effort volume.

Each attribute receives a rating on a six point scale which usually ranges between very low categories going up to extra high category. The following table shows an effort multiplier that applies to the rating. Table 4.6 illustrates the six point scale rating for each cost drivers.

### Table 4.6 Six Point Scale Rating

<table>
<thead>
<tr>
<th>Sno.</th>
<th>Cost Drivers</th>
<th>Ratings</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>RELY</td>
<td>0.75</td>
<td>0.88</td>
<td>1.00</td>
<td>1.15</td>
<td>1.40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>DATA</td>
<td>0.94</td>
<td>1.00</td>
<td>1.08</td>
<td>1.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>CPLX</td>
<td>0.70</td>
<td>0.85</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td>1.65</td>
<td></td>
</tr>
<tr>
<td><strong>Hardware Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>TIME</td>
<td></td>
<td>1.00</td>
<td>1.11</td>
<td>1.30</td>
<td>1.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>STOR</td>
<td></td>
<td>1.00</td>
<td>1.06</td>
<td>1.21</td>
<td>1.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>VIRT</td>
<td>0.87</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>TURN</td>
<td>0.87</td>
<td>1.00</td>
<td>1.07</td>
<td>1.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Personnel Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>ACAP</td>
<td>1.46</td>
<td>1.19</td>
<td>1.00</td>
<td>0.86</td>
<td>0.71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>AEXP</td>
<td>1.29</td>
<td>1.13</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>PCAP</td>
<td>1.42</td>
<td>1.17</td>
<td>1.00</td>
<td>0.86</td>
<td>0.70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>VEXP</td>
<td>1.21</td>
<td>1.10</td>
<td>1.00</td>
<td>0.90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>LEXP</td>
<td>1.14</td>
<td>1.07</td>
<td>1.00</td>
<td>0.95</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Project Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>MODP</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>TOOL</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>SCED</td>
<td>1.23</td>
<td>1.08</td>
<td>1.00</td>
<td>1.04</td>
<td>1.10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The intermediate COCOMO model is represented as follows:

\[
E = a_i(KLOC)^{b_i}(EAF)
\]

(4.18)

Here, E denotes the effort applied for person-month,

KLOC denotes the estimated number of thousands of delivered LOC,
EAF denotes the factor estimated,

\[ a_i \] is the coefficient and \[ b_i \] is the exponent.

The development time estimation is similar to the basic COCOMO model.

4.7 DETAILED COCOMO MODEL IN E-LEARNING PROJECTS

The detailed COCOMO model includes all the properties of the intermediate version with the impact of cost drivers on each step of the software engineering process. This model uses the different effort multipliers applied for each cost driver attribute. It estimates the amount of effort needed to complete the each phase. Here, the e-learning project is divided into different modules and the COCOMO model is applied to determine the effort and then sums up the effort. The effort of the e-learning project is estimated based on the project size and cost drivers according the software life cycle. The five phases of this COCOMO model are:

- Plan and Requirement
- System design
- Detailed design
- Module code and test
- Integration and test

4.8 SUMMARY

In the proposed method, the three types of COCOMO models are applied on the e-learning project to determine the efforts required for project completion. The experimentation results and analysis on the e-learning project outcomes are explained in the next chapter.