**Practical No. 3**

**Aim: Estimation of Project Metrics.**

**Theory:**

A software project is not just about writing a few hundred lines of source code to achieve a particular objective. The scope of a software project is comparatively quite large, and such a project could take several years to complete. As in any other science and engineering discipline, one would be interested to measure how complex a project is. One of the major activities of the project planning phase, therefore, is to estimate various project parameters in order to take proper decisions.

**Some important project parameters that are estimated include:**

**Project size:** What would be the size of the code written say, in number of lines, files, modules?

**Cost:** How much would it cost to develop a software? A software may be just pieces of code, but one has to pay to the managers, developers, and other project personnel.

Duration: How long would it be before the software is delivered to the clients?

**Effort:**How much effort from the team members would be required to create the software?

**Two methods for estimating project metrics: COCOMO and Halstead's method.**

**COCOMO**

COCOMO (Constructive Cost Model) was proposed by Boehm. According to him, there could be three categories of software projects:

.

**1. Organic:** A development project is said to be of organic type, if

1)The project deals with developing a well understood application

2)The development team is small

3)The team members have prior experience in working with similar types of projects

**2. Semidetached**: A development project can be categorized as semidetached type, if

1)The team consists of some experienced as well as inexperienced staff

2)Team members may have some experience on the type of system to be developed

**3. Embedded:** Embedded type of development project are those, which

1)Aims to develop a software strongly related to machine hardware

2)Team size is usually large

Boehm suggested that estimation of project parameters should be done through three stages: Basic COCOMO, Intermediate COCOMO, and Detailed COCOMO.

1. **Basic COCOMO Model**

The basic COCOMO model helps to obtain a rough estimate of the project parameters. It estimates effort and time required for development in the following way:  
Effort = a \* (KLOC)b PM

Tdev = 2.5 \* (Effort)c Months

Where,

KLOC is the estimated size of the software expressed in Kilo lines of code

a, b, c are constants determined by the category of software project

Effort denotes the total effort required for the software development, expressed in person months (PMs)

Tdev denotes the estimated time required to develop the software (expressed in months)

The value of the constants a, b, c are given below:

| Software project | a | B | c |
| --- | --- | --- | --- |
| Organic | 2.4 | 1.05 | 0.38 |
| Semi-detached | 3.0 | 1.12 | 0.35 |
| Embedded | 3.6 | 1.20 | 0.32 |

1. **Intermediate COCOMO Model**

The basic COCOMO model considers that effort and development time depends only on the size of the software. However, in real life there are many other project parameters that influence the development process. The intermediate COCOMO take those other factors into consideration by defining a set of 15 cost drivers. Each of the 15 such attributes can be rated on a six-point scale ranging from "very low" to "extra high" in their relative order of importance.The product of effort multipliers of all the 15 attributes gives the Effort Adjustment Factor (EAF).

EAF is used to refine the estimates obtained by basic COCOMO as follows:

Effort\_corrected = Effort \* EAF

Dev\_corrected = 2.5 \* (Effort|\_corrected)c

1. **Detailed COCOMO Model**

Both the basic and intermediate COCOMO models consider a software to be a single homogeneous entity -- an assumption, which is rarely true. In fact, many real life applications are made up of several smaller sub-systems. The complete COCOMO model takes these factors into account to provide a far more accurate estimate of project metrics.

To illustrate this, consider a very popular distributed application: the ticket booking system of the Indian Railways. The ticket booking system has three main components:

Database, Graphical User Interface (GUI) and Networking facilities.

Among these, development of the GUI is considered as an organic project type; the database module could be considered as a semi-detached software. The networking module can be considered as an embedded software. To obtain a realistic cost, one should estimate the costs for each component separately, and then add it up.

* **Advantages of COCOMO**

COCOMO is a simple model, and should help one to understand the concept of project metrics estimation.

* **Drawbacks of COCOMO**

COCOMO uses KLOC, which is not a proper measure of a program's size. Indeed, estimating the size of a software is a difficult task.

**Case Study**

**A Library Information System**

Using COCOMO and based on the team size (small) and experience (high), the concerned project could be categorized as "organic". The experts, based on their prior experience, suggested that the project size could roughly be around 10 KLOC. This would serve as the basis for estimation of different project parameters using basic COCOMO, as shown below:

Effort = a \* (KLOC)b PM

Tdev = 2.5 \* (Effort)c Months

For organic category of project the values of a, b, c are 2.4, 1.05, 0.38 respectively. So, the projected effort required for this project becomes

Effort = 2.4 \* (10)1.05 PM

= 27 PM (approx)

So, around 27 person-months are required to complete this project. With this calculated value for effort we can also approximate the development time required:

Tdev = 2.5 \* (27)0.38 Months

= 8.7 Months (approx)

So, the project is supposed to be complete by nine months. However, estimations using basic COCOMO are largely idealistic. Let us refine them using intermediate COCOMO. Before doing so we determine the Effort Adjustment Factor (EAF) by assigning appropriate weight to each of the following attributes.

| **Cost Drivers** | **Ratings** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** |
| **Product attributes** |  |  |  |  |  |  |
| Required software reliability | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 |  |
| Size of database |  | 0.94 | 1.00 | 1.08 | 1.16 |  |
| Complexity of the product | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 | 1.65 |
| **Hardware attributes** |  |  |  |  |  |  |
| Run-time performance constraints |  |  | 1.00 | 1.11 | 1.30 | 1.66 |
| Memory constraints |  |  | 1.00 | 1.06 | 1.21 | 1.56 |
| Volatility of the virtual machine environment |  | 0.87 | 1.00 | 1.15 | 1.30 |  |
| Required turnabout time |  | 0.87 | 1.00 | 1.07 | 1.15 |  |
| **Personnel attributes** |  |  |  |  |  |  |
| Analyst capability | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 |  |
| Applications experience | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 |  |
| Software engineer capability | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 |  |
| Virtual machine experience | 1.21 | 1.10 | 1.00 | 0.90 |  |  |
| Programming language experience | 1.14 | 1.07 | 1.00 | 0.95 |  |  |
| **Project attributes** |  |  |  |  |  |  |
| Application of software engineering methods | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 |  |
| Use of software tools | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 |  |
| Required development schedule | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 |  |

The highlighted cells highlight our choice of weight for each of the cost drivers. EAF is determined by multiplying all the chosen weights. So, we get

EAF = 0.53 (approx)

Using this EAF value we refine our estimates from basic COCOMO as shown below

Effort|corrected = Effort \* EAF

= 27 \* 0.53

= 15 PM (approx)

Tdev|corrected = 2.5 \* (Effort|corrected)c

= 2.5 \* (15)0.38

= 7 months (approx)

After refining our estimates it seems that seven months would likely be sufficient for completion of this project. This is still a rough estimate since we have not taken the underlying components of the software into consideration. Complete COCOMO model considers such parameters to give a more realistic estimate.

**Halstead's Complexity Metrics**

Halstead took a linguistic approach to determine the complexity of a program. According to him, a computer program consists of a collection of different operands and operators. Halstead's metrics are computed based on the operators and operands used in a computer program. Any given program has the following four parameters:

n1: Number of unique operators used in the program

n2: Number of unique operands used in the program

N1: Total number of operators used in the program

N2: Total number of operands used in the program

Using the above parameters one compute the following metrics:

Program Length: N = N1 + N2

Program Vocabulary: n = n1 + n2

Volume: V = N \* lg n

Difficulty: D = (n1 \* N2) / (2 \* n2)

Effort: E = D \* V

Time to Implement: T = E / 18 (in seconds)

Example:

if(x>5)

{

x=x+2;

if(x<7)

{

x=0;

}

}

|  |  |
| --- | --- |
| operator | Operand |
| if | x |
| () | 5 |
| > | x |
| { | x |
| = | 2 |
| ; | x |
| + | 7 |
| if | x |
| () | 0 |
| < |  |
| { |  |
| = |  |
| ; |  |
| } |  |
| } |  |

n=Distinct N=Total 1=operators 2=operands

N1=15,N2=9;n1=9;n2=5

1. N=N1=N2;//length of the program

N=15+9=24

1. n=n1+n2; //vocabulary of program

n=9+5=14

1. purity Ratio=N̂/N

N̂ =n1log2n1+n2log2n2

N̂ =9log29+5log25

=40.13

N

N̂ /N=purity ratio=1.63

1. Volume: V = N \* log2 n

V=24log214

V=91.36

1. Difficulty: D = (n1 \* N2) / (2 \* n2)

D=(9\*9)/(2\*5)=8.1

1. Effort: E = D \* V

E=91.36\*8.1

E=740.016

The program volume V is the minimum number of bits needed to encode the program. It represents the size of the program while taking into account the programming language.  
The difficulty metric indicates how difficult a program is to write or understand.  
Effort denotes the "mental effort" required to develop the software, or to recreate the same in another programming language.

**Conclusion:** Thus we have studied project estimation techniques.