GOVERNMENT POLYTECHNIC PUNE

Name : Snehal Ganesh Dahake
En no : 1907011
Batch : A
Software Engineering
IT4101
==========

PRACTICAL 6

Aim :-

Estimate cost of project using COCOMO (constructive cost model)/COCOMO II Approach for the assigned project.

Theory :-

Measurement in Software Engineering :-

To assess the quality of the engineered product or system and to better understand the models that are created, some measures are used. These measures are collected throughout the software development life cycle with an intention to improve the software process on a continuous basis. Measurement helps in estimation, quality control, productivity assessment and project control throughout a software project. Also, measurement is used by software engineers to gain insight into the design and development of the work products. In addition, measurement assists in strategic decision-making as a project proceeds.

Software measurements are of two categories, namely, direct measures and indirect measures. Direct measures include software processes like cost and effort applied and products like lines of code produced, execution speed, and other defects that have been reported. Indirect measures include products like functionality, quality, complexity, reliability, maintainability, and many more.

Measurement process is characterized by a set of five activities, which are listed below.
Formulation: This performs measurement and develops appropriate metrics for software under consideration.

- **Collection :-**
  - This collects data to derive the formulated metrics.
- **Analysis :-**
  - This calculates metrics and the use of mathematical tools.
- **Interpretation :-**
  - This analyses the metrics to attain insight into the quality of representation.
- **Feedback :-**
  - This communicates recommendations derived from product metrics to the software team.
- **Lines of Code (LOC) :-**
  - The line of code (LOC) metric is any line of text in a code that is not a comment or blank line, in any case of the number of statements or fragments of statements on the line. LOC clearly consists of all lines containing program header files, declaration of any variable, and executable and non-executable statements. As Lines of Code (LOC) only counts the proportion of code, you can only utilize it to compare or estimate projects that utilize the same language and are programmed using the same coding standards.

Example :-

```
void selSort(int x[], int n) {
  //Below function sorts an array in ascending order
   int i, j, min, temp;
   for (i = 0; i < n - 1; i++) {
      min = i;
      for (j = i + 1; j < n; j++)
      if (x[j] < x[min])
      min = j;
      temp = x[i];
      x[i] = x[min];
      x[min] = temp;
   }
}
```

So, now If LOC is simply a count of the numbers of lines, then the above function shown contains 13 lines of code. But when comments and blank lines are ignored, the function shown above contains 12 lines of code (LOC).

Function Point (FP) :-

In the function point metric, the number and type of functions held up by the software are used to find FPC (function point count).

Calculating Function Point and Formula :-

Step-1 :-

F = 14 * scale

Scale varies from 0 to 5 according to the character of Complexity Adjustment Factor (CAF). Below table shows scale:

0 - No Influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 - Essential

Step-2 :-

Calculate Complexity Adjustment Factor (CAF).

CAF = 0.65 + (0.01 * F)

Step-3 :-

Calculate Unadjusted Function Point (UFP).

Function Units	Low	Average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Multiply each individual function point to corresponding values in the table.

Step-4 :-
Calculate Function Point.
FP = UFP * CAF

Example:
Given the following values, compute function points when all complexity adjustment factors (CAF) and weighting factors are average.

User Input = 50
User Output = 40
User Inquiries = 35
User Files = 6
External Interface = 4

Explanation :-

Step-1 :-
As complexity adjustment factor is average, hence, scale = 3.
F = 14 * 3 = 42

Step-2 :-
CAF = 0.65 + (0.01 * 42) = 1.07

Step-3 :-
As weighting factors are also average hence, we will multiply each individual function point to corresponding values in TABLE.
UFP = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628

Step-4 :-
Function Point = 628 * 1.07 = 671.96
This is the required answer.

COCOMO Model (Constructive Cost Model) :-

COCOMO (Constructive Cost Model) is a regression model based on LOC, that is the number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.

The key parameters which define the quality of any software products, which are also an outcome of the COCOMO are primarily Effort & Schedule :-

- ➢ **Effort :-**
  - Amount of labor that will be required to complete a task. It is measured in person-months units.
- ➢ **Schedule :-**
  - Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Types of Models :-

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements.

These are types of COCOMO model :-

1.Basic COCOMO Model :-

Basic COCOMO can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

Estimation of effort for calculating in Basic Model :-

E= a(KLOC)^b
time= c(Effort)^d
Person required = Effort/ time

The above formula is used for the cost estimation of the basic COCOMO model, and also is used in the subsequent models. The constant values a, b, c and d for the Basic **Model for the different categories of system :-**

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. The development time is measured in Months.

2. Intermediate COCOMO Model :-

The basic COCOMO model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability.

Formula :-

E= (a(KLOC)^b)*EAF

The values of a and b in the case of the intermediate model are as follows :-

Software Projects	a	b
Organic	3.2	1.05
Semi-Detached	3.0	1.12
Embedded	2.8	1.20

3. Detailed Model :-

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed COCOMO, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are :-

- ➢ Planning and requirements
- ➢ System design
- ➢ Detailed design
- ➢ Module code and test
- ➢ Integration and test
- ➢ Cost Constructive model

Conclusion :-

Hence, we have successfully estimated the cost of the project using the COCOMO (constructive cost model)/COCOMO II Approach for the assigned project.

==========

Name : Snehal Ganesh Dahake
En no : 1907011
Batch : A
Software Engineering
IT4101