**EXPERIMENT 6**

**DATE: 28/01/2020**

**[Estimation of Project Metrics](http://vlabs.iitkgp.ernet.in/se/2/)**

* Function Point Analysis | COCOMO Model

**ESTIMATION OF PROJECT METRICS**

Estimation of the size of software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project. Various measures are used in project size estimation. Some of these are:

* Lines of Code
* Number of entities in ER diagram
* Total number of processes in detailed data flow diagram
* Function points

**1. Lines of Code (LOC):** As the name suggest, LOC count the total number of lines of source code in a project. The units of LOC are:

* KLOC - Thousand lines of code
* NLOC - Non comment lines of code
* KDSI - Thousands of delivered source instruction

The size is estimated by comparing it with the existing systems of same kind. The experts use it to predict the required size of various components of software and then add them to get the total size.

**Advantages:**

* Universally accepted and is used in many models like COCOMO.
* Estimation is closer to developer’s perspective.
* Simple to use.

**Disadvantages:**

* Different programming languages contains different number of lines.
* No proper industry standard exists for this technique.
* It is difficult to estimate the size using this technique in early stages of project.

**2. Number of entities in ER diagram:**ER model provides a static view of the project. It describes the entities and its relationships. The number of entities in ER model can be used to measure the estimation of size of project. Number of entities depends on the size of the project. This is because more entities needed more classes/structures thus leading to more coding.

**Advantages:**

* Size estimation can be done during initial stages of planning.
* Number of entities is independent of programming technologies used.

**Disadvantages:**

* No fixed standards exist. Some entities contribute more project size than others.
* Just like FPA, it is less used in cost estimation model. Hence, it must be converted to LOC.

**3. Total number of processes in detailed data flow diagram:**

Data Flow Diagram (DFD) represents the functional view of a software. The model depicts the main processes/functions involved in software and flow of data between them. Utilization of number of functions in DFD to predict software size. Already existing processes of similar type are studied and used to estimate the size of the process. Sum of the estimated size of each process gives the final estimated size.

**Advantages:**

* It is independent of programming language.
* Each major process can be decomposed into smaller processes. This will increase the accuracy of estimation

**Disadvantages:**

* Studying similar kind of processes to estimate size takes additional time and effort.
* All software projects are not required to construction of DFD.

**4. Function Point Analysis:** In this method, the number and type of functions supported by the software are utilized to find FPC (function point count). The steps in function point analysis are:

* Count the number of functions of each proposed type.
* Compute the Unadjusted Function Points (UFP).
* Find Total Degree of Influence (TDI).
* Compute Value Adjustment Factor (VAF).
* Find the Function Point Count (FPC).

The explanation of above points given below:

* **Count the number of functions of each proposed type:**

Find the number of functions belonging to the following types:

* + External Inputs: Functions related to data entering the system.
  + External outputs: Functions related to data exiting the system.
  + External Inquiries: They leads to data retrieval from system but don’t change the system.
  + Internal Files: Logical files maintained within the system. Log files are not included here.
  + External interface Files: These are logical files for other applications which are used by our system.

**Compute the Unadjusted Function Points (UFP):**

| **FUNCTION TYPE** | **SIMPLE** | **AVERAGE** | **COMPLEX** |
| --- | --- | --- | --- |
| External Inputs | 3 | 4 | 6 |
| External Output | 4 | 5 | 7 |
| External Inquiries | 3 | 4 | 6 |
| Internal Logical Files | 7 | 10 | 15 |
| External Interface Files | 5 | 7 | 10 |

Categorise each of the five function types as simple, average or complex based on their complexity. Multiply count of each function type with its weighting factor and find the weighted sum. The weighting factors for each type based on their complexity are as follows:

**Find Total Degree of Influence:**

 Use the ’14 general characteristics’ of a system to find the degree of influence of each of them. The sum of all 14 degrees of influences will give the TDI. The range of TDI is 0 to 70. The 14 general characteristics are: Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, On-Line Data Entry, End-user Efficiency, Online Update, Complex Processing Reusability, Installation Ease, Operational Ease, Multiple Sites and Facilitate Change.  
Each of above characteristics is evaluated on a scale of 0-5.

**Compute Value Adjustment Factor (VAF):**

\ Use the following formula to calculate VAF  
VAF = (TDI \* 0.01) + 0.65

**Find the Function Point Count:**

 Use the following formula to calculate FPC  
FPC = UFP \* VAF

**Advantages:**

* It can be easily used in the early stages of project planning.
* It is in depending on the programming language.
* It can be used to compare different projects even if they use different technologies (database, language etc).

**Disadvantages:**

* It is not good for real time systems and embedded systems.
* Many cost estimation models like COCOMO uses LOC and hence FPC must be converted to LOC.

**COCOMO**  (Constructive Cost Model) is a regression model based on LOC, i.e **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. It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best-documented models.

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.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations.

Boehm’s definition of semidetached system:

1. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.

The above system types utilize different values of the constants used in Effort Calculations.

E = 3.0\*(400) ^ 1.12

= 2462.79PM

D = 2.5(2462.79) ^ 0.35

=38.45

**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
  2. Intermediate COCOMO Model
  3. Detailed COCOMO Model

The first level, **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.

**Intermediate COCOMO**takes these Cost Drivers into account and **Detailed COCOMO** additionally accounts for the influence of individual project phases, i.e in case of Detailed it accounts for both these cost drivers and also calculations are performed phase wise henceforth producing a more accurate result. These two models are further discussed below.

**Estimation of Effort: Calculations**

**1.Basic Model**

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

| **SOFTWARE PROJECTS** | **A** | **B** |
| --- | --- | --- |
| Organic | 2.4 | 1.05 |
| Semi Detached | 3.0 | 1.12 |
| Embedded | 3.6 | 1.20 |

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, expertise is taken into account, henceforth the estimate is rough.

**2.Intermediate 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 system. 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. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation.

Classification of Cost Drivers and their attributes:

**(i) Product attributes**

* + - Required software reliability extent
    - Size of the application database
    - The complexity of the product

**(ii) Hardware attributes**

* + - Run-time performance constraints
    - Memory constraints
    - The volatility of the virtual machine environment
    - Required turnabout time

**(iii) Personnel attributes –**

* + - Analyst capability
    - Software engineering capability
    - Applications experience
    - Virtual machine experience
    - Programming language experience

**(iv) Project attributes**

* + - Use of software tools
    - Application of software engineering methods
    - Required development schedule

| **COST DRIVERS** | **VERY LOW** | **LOW** | **NOMINAL** | **HIGH** | **VERY HIGH** |
| --- | --- | --- | --- | --- | --- |
| **Product Attributes** |  |  |  |  |  |
| Required Software Reliability | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 |
| Size of Application Database |  | 0.94 | 1.00 | 1.08 | 1.16 |
| Complexity of The Product | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 |
| **Hardware Attributes** |  |  |  |  |  |
| Run time Performance Constraints |  |  | 1.00 | 1.11 | 1.30 |
| Memory Constraints |  |  | 1.00 | 1.06 | 1.21 |
| Volatility of the virtual machine environment |  | 0.87 | 1.00 | 1.15 | 1.30 |
| Required turnabout time |  | 0.94 | 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 project manager is to rate these 15 different parameters for a particular project on a scale of one to three. Then, depending on these ratings, appropriate cost driver values are taken from the above table. These 15 values are then multiplied to calculate the EAF (Effort Adjustment Factor). The Intermediate COCOMO formula now takes the form:

**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

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software life cycle.

|  |  |  |  |
| --- | --- | --- | --- |
| **Presentation (4)** | **Documentation (3)** | **Explanation (3)** | **Total (10)** |
|  |  |  |  |