ISSN No: - 2456- 2165

Review of COCOMO and BAT-COCOMO Model

H.John, Department of Electrical And Computer Engineering,
Niger State, Nigeria.

***Abstract* – Nowadays there are many models for software development cost estimation, providing project managers with helpful information to make the right decisions. One of such well known mathematical models is the COCOMO model. To estimate costs and time, this model uses coefficients, which were determined in 1981 by means of the regression analysis of statistical data based on 63 different types of project data. Using these coefficients for a modern project, the appraisal may not be accurate; therefore, the aim of this paper is to optimize the model coefficients with genetic algorithms. Genetic algorithms are evolutionary methods for optimization. To evaluate population, the genetic algorithm will use a set of descriptive attributes of several software development projects. These attributes are the number of lines of a code, costs and implementation time of a project. Project costs estimated by means of the COCOMO model will be compared with the real ones, this way evaluating the fitness of an individual in the population of possible solutions.**

***Keywords* –** *COCOMO model, genetic algorithm, and software cost estimation.*

I. INTRODUCTION

Software cost estimation is essential for software project management. Accurate software estimation can provide good support for the decision-making process like the accurate assessment of costs can help the organization to better analyse the project and effectively manage the software development process, thus significantly reducing the risk. Once the planning is too pessimistic, it may lose business opportunities, but too optimistic planning can cause significant loss. There are several software cost estimation models to help project managers to make the right decisions. One of such models is the COCOMO model (Constructive Cost Model). It was introduced in 1981 by Barry Boehm – the famous scientist who contributed to the development of software project management by creating a scientific approach. The COCOMO model is based on 63 different types of statistical data analysis project. The actual number of lines of code, amount of effort and time were estimated and some coefficients, which depend on the software project, were developed and identified during the regression analysis phase.

Today's project evaluation based on old coefficients may not match the required accuracy; therefore, the aim of this research is to optimize the model coefficients [4]. The COCOMO model has three modes, depending on the size of the project and the project team size. The model has three levels. The accuracy of the base level is lower than in the intermediate level and detailed level because the estimation of effort uses only actual amount and information about the mode and does not use cost drivers, which include a subjective judgment of the product, project, personnel and hardware characteristics. Thus, in this article a basic level of COCOMO model will be discussed [4], [6].

Genetic algorithms are optimization algorithms in evolutionary computing techniques, proposed in 1975 by a scientist Holland. It is a natural heuristic algorithm that is used to find the exact and approximate solutions. Algorithm is based on the iterative improvement of the current solution, but a solution set is used instead of one solution. Most genetic algorithm applications are linked to a large-scale information processing and the development of prediction models [3].

II.BACKGROUND

A brief description of the classic COCOMO model is given in the present section.

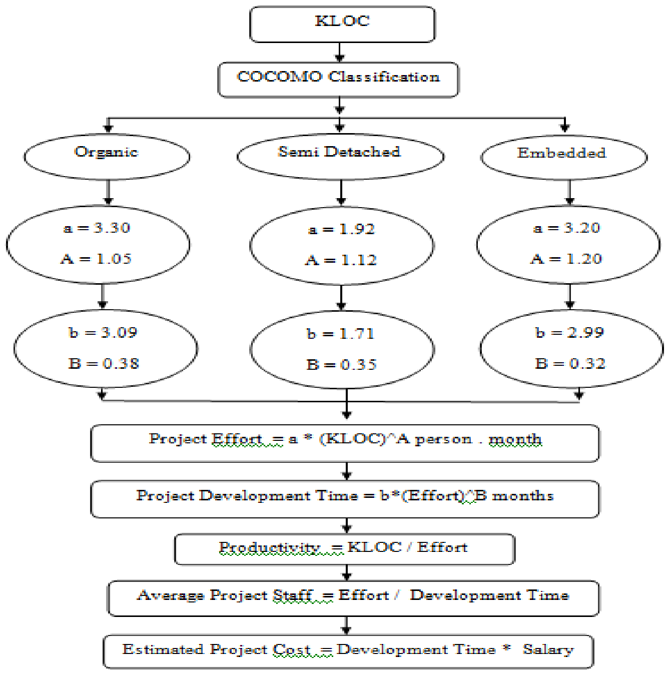
KLOC
COCOMO Classification
Organic
Semi Detached
Embedded
a = 3.30
A = 1.05
a = 1.92
A = 1.12
a = 3.20
A = 1.20
b = 3.09
B = 0.38
b = 1.71
B = 0.35
b = 2.99
B = 0.32
Project Effort = a * (KLOC)^A person . month
Project Development Time = b*(Effort)^B months
Productivity = KLOC / Effort
Average Project Staff = Effort / Development Time
Estimated Project Cost = Development Time * Salary

Fig 1. COCOMO Model

A. COCOMO Model

The COCOMO model has three modes to classify complexity of the system. The COCOMO model has three levels, providing increase of accuracy in each subsequent level. To calculate the effort at the base level, the equation (1) is used. It shows that effort is linearly dependant on the project size and rapidly changes if there is another mode [7], [9]. To evaluate project development time, the equation (2) is used.

$$E = a \,.\, KLOC^b$$

a, b – the COCOMO model coefficients;
$KLOC$ – the kilo-lines of code;
E – the effort (man-months)

II. BAT ALGORITHM

In this algorithm search is motivated by social behavior of bats and phenomenon of echolocation. It is a novel meta-heuristic technique for global numerical optimization problems. BA is used to optimize the weights of the parameters. These optimized weights can then be used for test effort estimation of new projects of a similar kind. In Bat algorithm, the spot of each bat is defined by and velocity, frequency, intensity, and the emission pulse rate in a D-dimensional search space. The two factors loudness and rate of pulse emission, i.e., A, r are also initialized with a constant value of 0 and 0.63 while F_{min} and F_{max} initialize by the 0.3292 and 0.9843.The value of α and β depend on the effort estimation and measure effort while the value of γ can be initialize by the 1.45.

The loudness is inversely proportional to the solution and the rate of pulse emission is directly proportional. Generate local solutions Y (t) and velocities V (t) at time step t by

$$F = F_{min} (F_{max} – F_{min})$$
$$V(t) = V(t-1) + (Y(t) – Y^*)$$
$$Y(t) =Y(t-1) + V(t)$$

III. ESTIMATION METHODS

All the cost estimation methods are based upon some form of analogy: Historical Analogy, Expert Judgment, Models, etc., the role these methods play in generating an estimate depends upon where one is in the overall life-cycle.

Historical analogy estimation methods are based upon using the software size, cost or effort of a comparable project from the past. When the term analogy is used in this document, the comparison is made using measures or data that has been recorded from completed software projects. Analogical estimates can be made at high levels using total software project size and/or cost for individual Work Breakdown Structure (WBS) categories in the process of developing the main software cost estimate. Generally, it is necessary to adjust the size or cost of the historical project, as there is rarely a perfect analogy. This is especially true for high-level analogies. [10]Analogy models are the simplest type of estimating models. They are used to estimate cost by comparing one program with a similar past program or programs, thereby avoiding issues with expert judgment bias. The advantage of the analogy method is that it is based on experience. However, the method is limited because, in most instances, similar programs do not exist. 8The steps using estimating by analogy are, Characterizing the proposed project; Selecting the most similar completed projects whose characteristics have been stored in the historical data base' Deriving the estimate for the proposed project from the most similar completed projects by analogy.

The main advantages of this method are, The estimation are based on actual project characteristic data; The estimator's past experience and knowledge can be used which is not easy to be quantified; The differences between the completed and the proposed project can be identified and impacts estimated.

Using this method, we have to determine how best to describe projects. The choice of variables must be restricted to information that is available at the point that the prediction required. Possibilities include the type of application domain, the number of inputs, the number of distinct entities referenced, the number of screens and so forth.

Even once we have characterized the project, we have to determine the similarity and how much confidence can we place in the analogies. Too few analogies might lead to maverick projects being used; too many might lead to the dilution of the effect of the closest analogies. Finally, we have to derive an estimate for the new project by using known effort values from the analogous projects. Possibilities include means and weighted means which will give more influence to the closer analogies.

***Expert judgment*:** The [5] majority of research work carried out in the software cost estimation field has been devoted to algorithmic models. However, by an overwhelming majority, expert judgment is the most commonly used estimation method. In its crudest form the expert judgment method involves consultation with one or more local experts who are knowledgeable about the development environment or application domain to estimate the effort required to complete a software project.

The method relies heavily on the experience of their knowledge in similar development environments and historically maintained databases on completed projects and the accuracy of theses past projects. The advantages of this method are, The experts can factor in differences between past project experience and requirements of the proposed project; The experts can factor in project impacts caused by new technologies, architectures, applications and languages involved in the future project and can also factor in exceptional personnel characteristics and interactions, etc. The disadvantages include, This method cannot be quantified; It is hard to document the factors used by the experts or experts-group; Expert may be some biased, optimistic, and pessimistic, even though they have been decreased by the group consensus; The expert judgment method always compliments the other cost estimating methods such as algorithmic method.

Delphi Approach or Wideband Delphi technique attempts to gather the opinions of a group of experts with the aim of producing an accurate unbiased estimate. It is a structured technique of expert judgment and is essentially a form based technique involving a multistep procedure: Experts are issued the specification and estimation form by the coordinator.

a. A group meeting is held to discuss the product and estimation issues.

b. Experts produce an independent estimate.

c. Estimates are returned indicating the median estimate and the expert's personal estimate.

d. Another group meeting is held to discuss results.

e. Experts prepare a revised independent estimate.

f. Steps 3-6 are repeated until a consensus is reached by the panel of experts.

Advantages of the Delphi Estimation Process:

a. 12Free of social pressure, personality influence, and individual dominance

b. Allows sharing of information and reasoning among participants

c. Conducive to independent thinking and gradual formulation

d. Respondent panel provides broad analytical perspective on problems and issues

e. Can be used to reach consensus among groups hostile towards each other

Disadvantages of the Delphi Estimation Process:

a. Judgments are those of a selected group, and may not represent prevailing opinion

b. Tendency to eliminate extreme positions and force middle-of-the-road consensus

c. More time-consuming than nominal group process

d. Requires skill in written communication

e. Requires adequate time and participant commitment (may require 30 to 45 days to complete entire process)

Algorithmic Method: The algorithmic method is designed to provide some mathematical equations to perform software estimation. These mathematical equations are based on research and historical data and use inputs such as Source Lines of Code (SLOC), number of functions to perform, and other cost drivers such as language, design methodology, skill-levels, risk assessments, etc. The algorithmic methods have been largely studied and there are a lot of models have been developed, such as COCOMO models, Putnam model, and function points based models. General advantages of methods are, it is able to generate repeatable estimations; It is easy to modify input data, refine and customize formulas; It is efficient and able to support a family of estimations or a sensitivity analysis; It is objectively calibrated to previous experience. General disadvantages are, It is unable to deal with exceptional conditions, such as exceptional personnel in any software cost estimating exercises, exceptional teamwork, and an exceptional match between skill-levels and tasks; Poor sizing inputs and inaccurate cost driver rating will result in inaccurate estimation; Some experience and factors cannot be easily quantified.

Bottom-up approach, each component of the software system is separately estimated and the results aggregated to produce an estimate for the overall system. The requirement for this approach is that an initial design must be in place that indicates how the system is decomposed into different components. The advantages of this methods are, It permits the software group to handle an estimate in an almost traditional fashion and to handle estimate components for which the group has a feel; It is more stable because the estimation errors in the various components have a chance to balance out. The disadvantages are, It may overlook many of the system-level costs (integration, configuration management, quality assurance, etc.) associated with software development; It may be inaccurate because the necessary information may not available in the early phase. It tends to be more time-consuming; It may not be feasible when either time and personnel are limited.

Top-down approach is the opposite of the bottom-up method. An overall cost estimate for the system is derived from global properties, using either algorithmic or non-algorithmic methods. The total cost can then be split up among the various components. This approach is more suitable for cost estimation at the early stage. The advantages of this method are, It focuses on system-level activities such as integration, documentation, configuration management, etc., many of which may be ignored in other estimating methods and it will not miss the cost of system-level functions; It requires minimal project detail, and it is usually faster, easier to implement. The disadvantages are, It often does not identify difficult low-level problems that are likely to escalate costs and sometime tends to overlook low-level components; It provides no detailed basis for justifying decisions or estimates.

COCOMO (Constructive Cost Model) models[4] cost and schedule estimation model was originally published in [Boehm 1981]. It became one of most popular parametric cost estimation models of the 1980s. But COCOMO '81 along with its 1987 Ada update experienced difficulties in estimating the costs of software developed to new life-cycle processes and capabilities. The COCOMO II research effort was started in 1994 at USC to address the issues on non sequential and rapid development process models, reengineering, reuse driven approaches, object oriented approaches etc. COCOMO II was initially published in the Annals of Software Engineering in

1995 [Boehm *et al.* 1995]. The model has three sub models, Applications Composition, Early Design and Post-Architecture, which can be combined in various ways to deal with the current and likely future software practices marketplace. A primary attraction of the COCOMO models is their fully-available internal equations and parameter values. Over a dozen commercial COCOMO '81 implementations are available.

The models have been widely accepted in practice. In the COCOMOs, the code-size S is given in thousand LOC (KLOC) and Effort is in person-month. The basic COCOMO model is simple and easy to use. As many cost factors are not considered, it can only be used as a rough estimate.
COCOMO model is a regression model. It is based on the analysis of 63 selected projects. The primary input is KDSI. [11]The problems are: In early phase of system life-cycle, the size is estimated with great uncertainty value. So, the accurate cost estimate cannot be arrived at; For this reason, the recalibration is necessary.

IV. SOFTWARE ESTIMATION

Software project plans include estimates of cost, resources, product size, schedules, staffing levels, and key milestones. The software estimation process is discussed in the following steps for developing software estimates. Establishing this process early in the life-cycle will result in greater accuracy and credibility of estimates and a clearer understanding of the factors that influence software development costs. This process also provides methods for project personnel to identify and monitor cost and schedule risk factors. The estimation method described is based upon the use of:

a. Multiple estimates

b. Data-driven estimates from historical experience

c. Risk and uncertainty impacts on estimates

d. [8]Different estimation methods may use different data. This results in better coverage of the knowledge base for the estimation process. It can help to identify cost components that cannot be dealt with or were overlooked in one of the methods

e. Different viewpoints and biases can be taken into account and reconciled. A competitive contract bid, a high business priority to keep costs down, or a small market window with the resulting tight deadlines tends to have optimistic estimates. A production schedule established by the developers is usually more on the pessimistic side to avoid committing to a schedule and budget one cannot meet.

V. CONCLUSION

Software cost estimation is simple in concept, but difficult and complex in reality. The difficulty and complexity required for successful estimates exceed the capabilities of most software project managers. Manual estimates are not sufficient for large applications. This paper has presented an overview of some software estimation techniques, providing an overview of several popular estimation models currently available. The important lesson to take from this paper is that no one method or model should be preferred over all others. The search for reliable, accurate and low cost estimation methods must continue. Also, more studies are needed to improve the accuracy of cost estimate for maintenance projects. The conclusion is that no single technique is best for all situations, and that a careful comparison of the results of several approaches is most likely to produce realistic estimates.

References

[1]. “Handbook for Software Cost Estimation”, Prepared by Karen Lum, Michael Bramble, Jairus Hihn, John Hackney, Mori Khorrami and Erik.

[2]. http://www.ceh.nasa.gov/downloadfiles/ Web%20 Links/cost_hb_public-6-5.pdf.

[3]. "Software Cost Estimation” by Hareton Leung and Zhang Fan.

[4]. “Software Development Cost Estimation Approaches – A Survey”, Barry Boehm, Chris Abts, Sunita Chulani.

[5]. http://www.ecfc.u-net.com/cost/index.htm

[6]. http://www.cs.odu.edu/~zeil/cs451/ Lectures/ 04mgmt/costest/costest_htsu3.html#subsubsect:parkinson

[7]. Software Cost Estimation in 2002, Capers Jones, Software Productivity Research Inc., Artemis Management Systems.

[8]. http://www.computing.dcu.ie/~renaat/ca421/ LWu1.html.

[9]. http://www.levela.com/software_cost_ estimating_swdoc.htm

[10]. Software Development Cost Estimating Guidebook, Software Technology Support Center Cost Analysis Group.

[11]. The Comparison of the Software Cost Estimating Methods, *Liming W.*

[12]. http://www.cs.uwf.edu/~wilde/gump/ delphi.htm

[13]. http://www.compapp.dcu.ie/~renaat/ ca421/LWu1.html