Vivek Jaglan et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 3, Issue 1, March 2016, pp. 20-26

Apply Fuzzy Optimization in Proficient Managing COCOMO Model Cost Drivers

Vivek Jaglan

Department of CSE, Amity School of Eng. & Tech, Amity University. Manesar, Haryana

ABSTRACT: - The COCOMO II model was developed in 1995. It could overcome the limitations of calculating the costs for non-sequential, rapid development, reengineering and reuse models of software. It has 3 modules: Application composition, early design & Post architecture. In COCOMO II, the constant value b is replaced by 5 scale factors. Basic COCMO Model is good for quick, early, rough order of magnitude estimate of software cost. It does not account for differences in hardware constraints, personal Quality and experience, use of modern tools and techniques, and other project attribute known to have a significant influence on software cost, which limits its accuracy. For this purpose, in this paper fuzzy optimization is being applied to reduce the level of MRE error in the cost estimation process

Key words: COCOMO Model, Software estimation, Fuzzy optimization.

I. INTRODUCTION

The COCOMO (Constructive Cost Estimation Model) is proposed by **DR. Berry Boehm in 1981** and that's why it is also known as COCOMO'81. It is a method for evaluating the cost of a software package. According to him software cost estimation should be done through three stages:

1. Basic COCOMO Model
2. Intermediate COCOMO Model
3. Complete/Detailed COCOMO Model

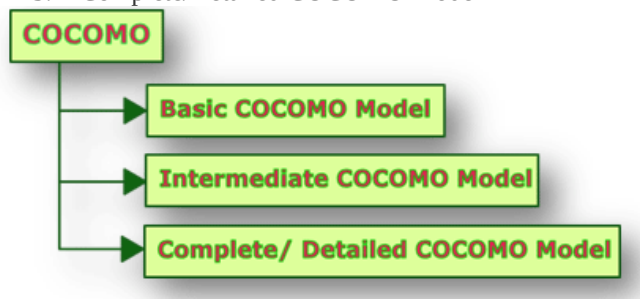

Figure 1 Type of Cocomo Model

COCOMO'81 models depend upon the two main equations:

1. **Development Effort :** MM = a * KDSI b
   Which is based on MM - man-month / person month / staff-month is one month of effort by one person. In COCOMO'81, there are 152 hours per Person month. According to organization this values may differ from the standard by 10% to 20%.
2. **Efforts and Development Time (TDEV) :** TDEV = 2.5 * MM c
   The coefficients a, b and c depend on the mode of the development [1].

DEVELOPMENT MODES:

There are three modes of development:

1. **Organic Mode:**
   - Relatively Small, Simple Software projects.
   - Small teams with good application experience work to a set of less than rigid requirements.
   - Similar to previously developed projects.
   - Relatively small and require little innovation.
2. **Semidetached Mode:**
   - Intermediate (in size and complexity) software projects in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements.
3. **Embedded Mode:**
   - Software projects that must be developed within set of tight hardware, software and operational Constraints [2].

Table 1 Development Mode with Project Characteristics:

	Size	**Innovation**	**Deadline**	**Dev. Environment**
ORGANIC	Small	Little	Not Tight	Stable
SEMI-DITACHED	Medium	Medium	Medium	Medium
EMBEDDED	Large	Greater	Tight	Complex Hardware

The values of a1, a2, b1, b2 for different categories of products (i.e. organic, semidetached, and embedded) as given by Boehm [1981] are summarized below. He derived the

above expressions by examining historical data collected from a large number of actual projects.

BASIC COCOMO MODEL: . It gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

Effort = a_1 x (KLOC)$^{a}{}_{2}$PM

Tdev = b_1 x (Effort)$^{b}{}_{2}$ Months

Where,

- KLOC is the estimated size of the software product expressed in Kilo Lines of Code,
- a_1, a_2, b_1, b_2 are constants for each category of software products,
- Tdev is the estimated time to develop the software, expressed in months,
- Effort is the total effort required to develop the software product, expressed in person months (PMs) [3].

Estimation of development effort:

For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic: **Effort = 2.4(KLOC)$^{1.05}$ PM**

Semi-Detached: **Effort = 3.0(KLOC)$^{1.12}$ PM**

Embedded: **Effort = 3.6(KLOC)$^{1.20}$ PM**

PM: Person Months

Estimation of development time:

For the three classes of software products, the formulas for estimating the development time based on the effort are given below:

Organic: **Tdev = 2.5(Effort)$^{0.38}$ Months**

Semi-detached: **Tdev = 2.5(Effort)$^{0.35}$ Months**

Embedded: **Tdev = 2.5(Effort)$^{0.32}$ Months**

The effort estimation is expressed in units of person-months (PM). It is the area under the person-month plot as shown in figure below. It should be carefully noted that an effort of 100 PM does not imply that 100 persons should work for 1 month nor does it imply that 1 person should be employed for 100 months, but it denotes the area under the person-mont curve.

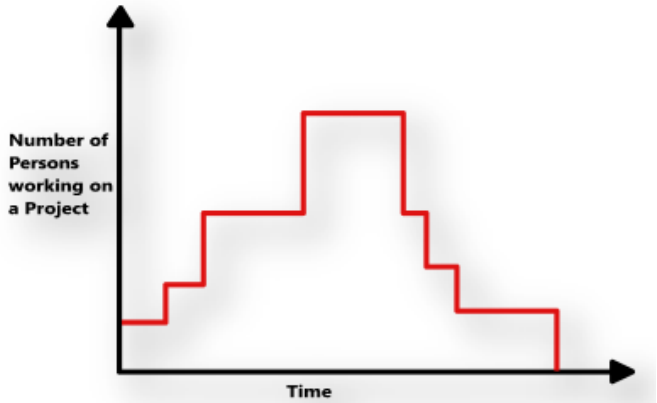

Fig. Person Month Curve

From the following figure which shows a plot of **estimated effort versus product size**. We can observe that the effort is somewhat superlinear in the size of the software product. Thus, the effort required to develop a product increases very rapidly with project size [5].

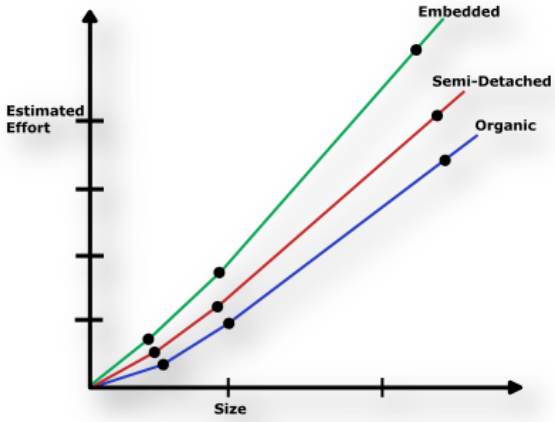

Fig. Effort Verses Product size

Now the following figure plots the **development time versus the product size in KLOC** can be observed that the development time is a sublinear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately.

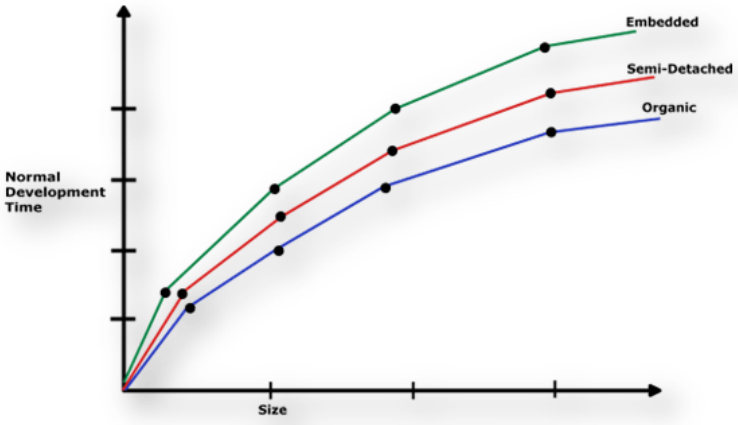

Fig. Development time Verses size

It is to be noted that the effort and the duration estimations obtained using the COCOMO model are called as nominal effort estimate and nominal duration estimate [6].

ADVANTAGES OF COCOMO MODEL:

- COCOMO is transparent, one can see how it works unlike other models such as SLIM
- Drivers are particularly helpful to the estimator to understand the impact of different factors that affect project costs.

LIMITATIONS OF COCOMO

- COCOMO is used to estimate the cost and schedule of the project, starting from the design phase and till

the end of integration phase. For the remaining phases a separate estimation model should be used.

- COCOMO is not a perfect realistic model. Assumptions made at the beginning may vary as time progresses in developing the project.
- When need arises to revise the cost of the project. A new estimate may show over budget or under budget for the project. This may lead to a partial development of the system, excluding certain requirements.
- COCOMO assumes that the requirements are constant throughout the development of the project; any changes in the requirements are not accommodated for calculation of cost of the project.
- There is not much difference between basic and intermediate COCOMO, except during the maintenance and development of the software project [7].
- COCOMO is not suitable for non-sequential, rapid development, reengineering, reuse cases models [8].
- It is hard to accurately estimate KDSI early on in the project, when most effort estimates are required.
- KDSI, actually, is not a size measure it is a length measure.
- Extremely vulnerable to mis-classification of the development mode.
- Success depends largely on tuning the model to the needs of the organization, using historical data which is not always available
- COCOMO model ignores requirements and all documentation.
- It ignores customer skills, cooperation, knowledge and other parameters.
- It oversimplifies the impact of safety/security aspects.
- It ignores hardware issues
- It ignores personnel turnover levels
- It is dependent on the amount of time spent in each phase [9].

II. COST ESTIMATION ACCURACY

The cost estimation may vary due to changes in the requirements, staff size, and environment in which the software is being developed.

The calculation for cost estimation accuracy is given as follows

$$\text{Absolute error} = (E_{pred} - E_{actual})$$

$$\text{Percentage error} = (E_{pred} - E_{actual})/E_{actual}$$

$$\text{Relative error} = 1/n \sum (E_{pred} - E_{actual})/E_{actual}$$

The above results give a more accurate estimation of costs for future projects [10]. The cost estimation model now becomes more realistic.

Effort (E) is calculated as follows

$$E = a * (KDSI)^{sf} * \pi (EM)$$

Where a is constant, sf is scaling factor, EM is Effort Multiplier (7 for Early design, 17 for Post architecture). It has following applications as given below:

- Helps in making decisions based on business and financial calculations of the project.
- Establishes the cost and schedule of the project under development, this provides a plan for the project.
- Provides a more reliable cost and schedule, hence the risk mitigation is easy to accomplish.
- It overcomes the problem of reengineering and reuse of software modules.
- Develops a process at each level. Hence takes care of the capability maturity model.[14]

III. RELATED WORK

Helmet. al (1992) correlated actual data with COCOMO estimated values and determinedif the COCOMO method accurately reflected documentedprogram expenditures. Because space bornemicroprocessing was a relatively new arena, the primary constraint associated withdeveloping a model was the limited available data base. It supported a statistical analyses is presented alongwith a discussion on calculated COCOMO results. In the analyses,the use of nonparametric statistics for small samples wasaddressed. Wilcoxon Signed-Rank and Kendall-Rank statisticssupported distribution free analyses of the data [15].

Leunget. al (2002) provided a general overview of software cost estimation methods including the recent advances in the field.As a number of these models relied on a software size estimate as input, they first provided anoverview of common size metrics. They then highlighted the cost estimation models that had beenproposed and used successfully. Models might be classified into 2 major categories: algorithmicand non-algorithmic. Each had its own strengths and weaknesses. A key factor in selecting a costestimation model was the accuracy of its estimates. Unfortunately, despite the large body ofexperience with estimation models, the accuracy of these models was not satisfactory. The paperincluded comment on the performance of the estimation models and description of several newerapproaches to cost estimation [16].

Huanget. al (2003) proposed novel neuro-fuzzy Constructive Cost Model(COCOMO) for software estimation. Themodel carried some of the desirable features of the neuro-fuzzy approach, such as learning ability and goodinterpretability, while maintaining the merits of theCOCOMO model. Unlike the standard neural networkapproach, this model was easily validated by experts andcapable of generalization. In addition, it allowed inputs to be continuous-rating values and linguistic values, therefore avoiding the problem of similar projects havingdifferent estimated costs. Also presented in this paper was adetailed learning algorithm. The validation, usingindustry project data, showed that the model

greatlyimproves the estimation accuracy in comparison with thewell-known COCOMO model [17].

Mbarkiet. al (2004)studied the use of FRBSs to provide a natural interpretationof cost estimation models based on a Back-propagation three-layer feedforward Perception. What they had proposed comprised essentially the use of the Benitez's method to extract the if-thenfuzzy rules from this network [I]. These fuzzy rules expressed the information encoded in thearchitecture of the network, and the interpretation of each fuzzy rule had been determined byanalyzing its premise and its output. While their case study had shown that they could explain themeaning of the output and the propositions composing the premise of each fuzzy rule, the entire fuzzy rule cannot be easily interpreted because it used the 'i-or' operator. In this paper, theyexploredanother mapping method, that is, the Jang and Sun method, to extract if-then fuzzy rules fromartificial neural networks. The use of this method requires that the architecture of the network be an RBFN [18].

Ismaeel et. al (2007)discussed the use of COCOMO II (Constructive CostModel) to estimate the cost of software engineering . The COCOMOII which allowed us estimate the cost, effort and scheduling whenplanning new software development. They used the effort equationguidance to find the number of person / months which was needed tocomplete the project and duration equation to specify the numbers ofmonths which was needed to complete this project.This paper presented how implemented COCOMO II modelequation about the same data in different language they chosen C andOOP(C++), they made the analysis of these two program and theyconcluded that relation between effort and duration was forwardrelation, they meant that when effort increasing duration would increasingin the author side [19].

Rollo et. al (2009) proposed an alternative use of the COCOMO model to assist in the task ofestimation. The generally accepted method of estimation using a functional sizing method was to base the estimate on previous project data, where those projects for a homogeneousset with the project under study. The chief difficulty was to find a sufficiently homogeneousset of projects. Research previously carried out can demonstrate that by increasing thedegree of homogeneity amongst a set of projects leads to a useful reduction in thevariation of the estimates. The proposal was that they might sensibly use the COCOMO costdrivers to allow us to determine a set of homogeneous projects by using a techniquederived from estimation by analogy. In addition the COCOMO cost drivers might be usedto allow the estimator to adjust his estimates based on the differences between the costdrivers exhibited by the available data and the project under study. This paper was theresult of ongoing research and it was offered as a position paper showing the resultsobtainable under research conditions. The author would be keen to establish links withpractitioners to undertake field trials of the proposed approach [20].

Živadinović et. al (2011) presented the most relevant methods and models for effort estimation used by software engineers in the past four decades. Classification of the methods had been alsosuggested as well as brief description of the estimation methods presented.In the past four decades a great number of different models and effort estimation methods had beendeveloped. This clearly indicated the awareness among the researchers of the need to improve effortestimation in software engineering. Unfortunately, the fact remains that even though, all the effortinvested by the researchers yielded no result as they wished for and, even today, effort estimation stillremained rather unreliable [21].

Al_Qmase et. al (2013) focused upon the COCOMO Model. It was further consisted of its two sub models called COCOMO I and COCOMO II. The primary objective of this research was to use an appropriate case study to evaluate the accuracy of the sub models COCOMO I and II and ascertain the variation of the realistic resource effort, staff and time. The findings to date showed that the Application Composition Model of COCOMO II was more accurate in determining time and cost for the successful conclusion of a software project than the other two COCOMO I and II Models for a similar application for example Task Manager [23].

Manikavelan et. al (2014) discussed the software cost estimation as an important factor for making estimation in software Engineering. The general question that came to their mind during cost estimation was how to make them accurate. This was a common area for failing the estimationlike the customers not sharing their requirements clearly andswift the technologies with payable to open source, and efficiency (memory and execution time) etcetera. No method was necessarily better or worse than the other, in fact, their strengths and weaknesses were often complimentary to each other. Non-Algorithmatic cost estimation contained the method named analogy. It helped to compare the proposed project to similarprojects from the past. In this paper feed forward neural algorithm was used in analogy to get accurate software cost estimation [24].

Kushwaha et. al (2014) proposed the software cost estimation model based on fuzzy logic. Software project costs included the cost incurred in all the expenses, i.e. the cost of projectfrom initiation, development to test, software management, qualitymanagement and contingent rework, etc. The imprecision inculcated from the inputs utilized in algorithmic models likeconstructive cost model COCOMO results in imprecise outputswhich leads to erroneous effort estimation The fuzzy logic model fuzzified the two parts of the COCOMO model i.e.nominal effort prediction and the effort adjustment factor. Theanalysis shows that the performance of the FIS enhanced byincreasing the number of membership functions. Validationexperiment was carried out on NASA 93 and COCOM08I public database [25].

Guptaet. al (2015) discussed a new calibrated Intermediate COCOMO model (for all types ofsystem i.e. organic, semi-detached and embedded) developed with Bat Algorithm and generated new optimizedcoefficients. Bat Algorithm was newest Algorithm amongst thecategory of Meta Heuristic and

population based Algorithms. For estimation they had employed Bat Algorithm on NASA 63dataset and results showed that optimized coefficients by Bat Algorithm gave better results in terms of MMRE (MeanMagnitude of Relative Error) for all types of projects ascompared to coefficients in COCOMO Model which wereobtained by Regression Analysis [26].

IV. COCOMO II COST DRIVER

The initial estimates made in the COCOMO II model are adjusted using a set of attributes (project cost drivers) that reflect:

1. Product characteristics such as the required system reliability and product complexity.
2. Computer characteristics such as execution time or memory constraints. These are constraints imposed on the software by the hardware platform.
3. Personnel characteristics such as programming language skills that take the experience and capabilities of the people working on the project into account.
4. Project characteristics of the software development project such as the IDE that is available and the development schedule.

The following table shows all 17 of the project cost drivers that may be taken into consideration.

The selection of scale drivers is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation. Each scale driver has a range of rating levels, from Very Low to Extra High. Each rating level has a weight, *W*, and the specific value of the weight is called a scale factor. A project's scale factors, *W*i, are summed across all of the factors, and used to determine a scale exponent, *B*, via the following formula:

V. FUZZY OPTIMIZATION BASED FRAMEWORK OF HANDLING COCOMO II COST DRIVER

For a COCOMO model to be accurate it must be calibrated using historical data. COCOMO 81 was calibrated using 63 data points from past projects. The calibration process can be done by using a company's own data, but for the most part it requires more data then a single company would have. The calibration involves doing a statistical analysis on your data and then adjusting all cost driver values.

Because of the need of a proper calibration there are standard calibrations released. COCOMO II has gone through two calibrations, COCOMO II.1997 and COCOMO II 1998. COCOMO II.1997 was based on 83 data points and was found that it only could come within 20% of the actual values 46% of the time. The COCOMO II.1998 calibration was found to come within 30% of the actual values 75% of the time, this calibration was based on 161 data points (Bohem, Chulani, Clark, 1997). Users can also submit data from their own projects to be used in future calibrations. When using the release calibrations or your own it is important to continue collecting historical data so it can be use to further increase the accuracy of your estimation results in the future.

Empirical software estimation models are mainly based on cost drivers and scale factors. These models show the problem of instability due to values of the cost drivers and scale factors, thus affecting the sensitivity in terms of accurate effort estimation. Also, most of the models depend on the size of the project and a small change in the size leads to the proportionate change in the effort. Miscalculations of the cost drivers have even more noisy data as a result too. For example, a misjudgment in personnel capability cost driver in COCOMO between "very high to very low" will result in 300% increase in effort. Similarly in SEER-SEM, changing security requirements values from "low" to "high" will result in 400% increase in effort. In PRICE-S, 20% change in effort will occur due to small change in the value of the productivity factor [14]. Above statements reveal that, all models have one or more inputs for which small changes will result in large changes in effort. The input data problem is further compounded in that some inputs are difficult to obtain, especially early stages in a program development. The size must be estimated early in a project using one or more sizing models. Some sensitive inputs, such as analyst and programmer capability in cost drivers, are based on individual and are often difficult to determine. Many studies like the one performed by [15] show that personnel parameter data are difficult to collect.

VI. MANAGING DRIVERS WITH FUZZY OPTIMIZATION

In an attempt to reduce the complexity inherent to the presence of severalobjectives, a fuzzy solution which is richer from an informational point of viewmay be desirable. Such a solution may be obtained via techniques of parametric programming (Chanas [14]). Advantages of using a fuzzy approach for finding a solution of a multipleobjective programming problem are flexibility, easiness to be adapted forinteractive use and the fact that such a methodology meets the main demands foroperational models: simplicity, robustness adaptively. The ideas outlined above have also been extended to the case when relevantdata are fuzzy parameters. Fuzzy set theory provides a host of attractive aggregation connectives for integrating membershipvalues representing uncertain information. These connectives can be categorized into thefollowing three classes union, intersection and compensation connectives.Union produces a high output whenever any one of the input values representing degreesof satisfaction of different features or criteria is high. Intersection connectives produce a highoutput only when all of the inputs have high values. Compensative connectives have the propertythat a higher degree of satisfaction of one of the criteria can compensate for a lowerdegree of satisfaction of another criteria to a certain extent. In the sense, union connectivesprovide full compensation and intersection connectives provide no compensation. In a decisionprocess the idea of trade-offs corresponds to viewing the global

evaluation of an action aslying between the worst and the best local ratings. This occurs in the presence of conflictinggoals, when a compensation between the corresponding capabilities are allowed. Averagingoperators realize trade-offs between objectives, by allowing a positive compensation betweenratings.

VII. CONCLUSION

Work carried out in the paper explores the inter-relationship among different dimensions of data driven software projects, namely, project size and effort. The above-mentioned results demonstrate that applying proposed method to the software effort estimation is by far the most feasible approach for addressing the problem of apprehension and ambiguity existing in software effort drivers. Order of occurrence of various cost drivers has a significant impact on overall efforts in project estimation. Small adjustments to the COCOMO cost drivers bring significant improvements to the quality criteria applied to the proposed approach. Proposed method is producing tuned values of the cost drivers, which are effective enough to improve the productivity of the projects. Prediction at different levels of MRE for each project reflects the percentage of projects with desired accuracy. Furthermore, this model is validated on two different datasets which represents better estimation accuracy as compared to the COCOMO 81 based NASA 63 and NASA 93 datasets. The utilization of proposed algorithm for other applications in the software engineering field can also be explored in the future.

VIII. References

[1]. K. M. Furulund and K. Moløkken-Østvold, "Increasing software effort estimation accuracy—using experience data, estimation models and checklists," in Proceedings of the 7th International Conference on Quality Software (QSIC '07), pp. 342–347, Portland, OR, USA, October 2007.

[2]. Q. Alam, P. Bhatia, and S. Sarwar, Systematic Review of Effort Estimation and Cost Estimation, Institute of Management Studies, Roorkee, India, 2012.

[3]. J. J. Dolado, On the Problem of the Software Cost Function, Facultad de Informatica, Universidad del Pais Vasco-EuskalHerrikoUnibertsitatea, Gipuzkoa, Spain, 2000.

[4]. K. Molokken and M. Jorgensen, "A review of software surveys on software effort estimation," in Proceedings of the International Symposium on Empirical Software Engineering (ISESE '03), pp. 220–230, 2003.

[5]. F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro, "Genetic programming for effort estimation: an analysis of the impact of different fitness functions," in Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10), pp. 89–98, IEEE Computer Society, DMI, University of Salerno, Benevento, Italy, October 2010.

[6]. A. F. Sheta, "Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects," Journal of Computer Science, vol. 2, no. 2, pp. 118–123, 2006.

[7]. B. W. Boehm, Software Engineering Economics, Prentice Hall, IEEE, 1984.

[8]. J. Magne and M. Shepperd, "A Systematic Review Of Software Development Cost Estimation Studies," IEEE Transactions on Software Engineering, vol. 33, no. 1, pp. 33–53, 2007.

[9]. P. L. Braga, A. L. I. Oliveira, and S. R. L. Meira, "A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation," in Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC '08), pp. 1788–1792, Ceará, Brazil, March 2008.

[10]. M. Harman and B. F. Jones, "Search-based software engineering," Information and Software Technology, vol. 43, no. 14, pp. 833–839, 2001.

[11]. J. Clarke, J. J. Dolado, M. Harman et al., "Reformulating software engineering as a search problem," IEE Proceedings: Software, vol. 150, no. 3, pp. 161–175, 2003.

[12]. M. Jørgensen and S. Grimstad, "Avoiding irrelevant and misleading information when estimating development effort," IEEE Software, vol. 25, no. 3, pp. 78–83, 2008.

[13]. A. L. Lederer and J. Prasad, "A causal model for software cost estimating error," IEEE Transactions on Software Engineering, vol. 24, no. 2, pp. 137–148, 1998.

[14]. S. Basha and P. Dhavachelvan, "Analysis of empirical software effort estimation models," International Journal of Computer Science and Information Security, vol. 7, no. 3, pp. 68–77, 2010.

[15]. Jim E. Helm, The Viability of Using Cocomo in the Special Application Software Bidding and Estimating, IEEE Transactions On Engineering Management, Vol. 39, No. 1, February 1992, pp. 42-58

[16]. Hareton Leung, Zhang Fan, Software Cost Estimation, In: Handbook Of Software Engineering And Knowledge Engineering, World Scientific Pub. Co, River Edge, Nj

[17]. Xishi Huang, Luiz F. Capretz, Jing Ren, A Neuro-Fuzzy Model for Software Cost Estimation, Proceedings of the Third International Conference On Quality Software (QSIC'03), 2003 IEEE,

[18]. Idri, A.; Mbarki, S.; Abran, A., "Validating and understanding software cost estimation models based on neural networks," Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on , vol., no., pp.433,434, 19-23 April 2004

[19]. Hana Rashied Ismaeel, Software Engineering Cost EstimationUsing COCOMO II Model, Al-Mansour Journal, Year: 2007 Issue: 10 Pages: 86-111

[20]. Dr Anthony L Rollo, Functional Size measurement and COCOMO – A synergistic approach, in Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '07), pp. 1256–1260, Singapore, December 2009.

[21]. Jovan Živadinović, Methods Of Effort Estimation In SoftwareEngineering, Proceeding International Symposium Engineering Management And Competitiveness 2011 (EMC2011)June 24-25, 2011, Zrenjanin, Serbia

[22]. Mohammed MugahedAl_Qmase, M. RizwanJameel Qureshi, Evaluation of the Cost Estimation Models: Case Study of Task Manager Application, International Journal of Modern Education and Computer Science, 2013, 8, 1-7

[23]. D.Manikavelan, Software Cost Estimation By Analogy Using Feed Forward Neural Network, Proceeding of ICICES2014 - S.A. Engineering College, Chennai, Tamil Nadu, India

[24]. Kushwaha, N.; Suryakant, "Software cost estimation using the improved fuzzy logic framework," IT in Business, Industry and Government (CSIBIG), 2014 Conference on , vol., no., pp.1,5, 8-9 March 2014,

[25]. Gupta, N.; Sharma, K., "Optimizing intermediate COCOMO model using BAT algorithm," Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on , vol., no., pp.1649,1653, 11-13 March 2015

[26]. B. L. Barber, Investigative search of quality historical software support cost data and software support cost-related data [M.S. thesis], 1991.

[27]. N. H. Chiu and S. J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," Journal of Systems and Software, vol. 80, no. 4, pp. 628–640, 2007.

[28]. G. Kadoda and M. Shepperd, "Using simulation to evaluate prediction techniques," in Proceedings of the 7th International Software Metrics Symposium (METRICS '01), pp. 349–359, IEEE Press, London, UK, 2001.

[29]. M. J. Shepperd and G. F. Kadoda, "Comparing software prediction techniques using simulation," IEEE Transactions on Software Engineering, vol. 27, no. 11, pp. 1014–1022, 2001.

[30]. M. J. Shepperd and C. Schofield, "Estimating software project effort using analogies," IEEE Transactions on Software Engineering, vol. 23, no. 11, pp. 736–743, 1997.

