Fuzzy Logic based framework for Software Development Effort Estimation

Sandeep Kad[1], Vinay Chopra[2]
[1]Department of Information Technology
Amritsar College of Engg. & Technology, Amritsar, Punjab, India
[2]Department of Computer Science
DAV Institute of Engg. & Technology, Jalandhar, Punjab, India
[1]kadsandeep@yahoo.com, [2]vinaychopra222@yahoo.co.in

Abstract- *Software development effort estimation is among one of the most challenging jobs that software developers need to perform. Due to the lack of information during the early stages of software development, the developers often express their inability to estimate accurately the effort, cost and schedule of the software under consideration. This inaccuracy in estimation leads to monetary losses as well delay in delivery of the product. In this paper, a soft computing based technique is explored to overcome the problems of uncertainty and imprecision resulting in improved process of software development effort estimation. In doing so, fuzzy logic is applied to different parameters of Constructive Cost Model (COCOMO) II. Results shows that the value of MMRE (Mean of Magnitude of Relative Error) and pred obtained by means of applying fuzzy logic is much better than of MMRE of algorithmic model. The validation of results is carried out on COCOMO dataset.*

Keywords- Software Cost Estimation, COCOMO, Soft Computing, Fuzzy Logic

I. INTRODUCTION

Software estimation accuracy is among the greatest challenges for software developers. It is a vital aspect that deals with planning and tracking of software project. Controlling the expenditure of software development effectively is of utmost importance in today's competitive environment [1]. The need for reliable and accurate software development cost predictions in software engineering is a challenging job as it accounts for considerable financial and strategic planning [2]. Software development estimation helps us in determining the effort, cost and schedule of developing the project under consideration. Despite considerable research and practical experience it is still a formidable challenge to understand and predict what happens in a large software projects. Even in cases where some parties have good understanding of consequences, the business pressures and lack of quantitative evidence often results in misguided effort and faulty plans [3]. Even sometimes, underestimation of project is done knowingly just to win contract which later on may result into monetary loses also a poor quality project being delivered to the client. An accurate cost estimation leads to effective control of time and budget during software development. To have an accurate estimate various models have been proposed in the past. Among those Boehm's COCOMO is the most commonly used because of its simplicity for estimating the effort in person-month for a project at different stages of development.

II. SOFTWARE EFFORT ESTIMATION MODELS

Cost estimation is one of the most challenging tasks in project management. Its purpose is to accurately estimate the resources needed and required schedules for software development projects. The software estimation process includes estimating the size of the software product to be produced, estimating the effort required, developing preliminary project schedules, and finally, estimating overall cost of the project [4]. However, the process estimation is uncertain in nature as it largely depends upon some attributes that are quite unclear during the early stages of development, but it needs to be carried out as huge investments are involved in developing the software [5]. Software effort estimation models are divided into two main categories: algorithmic models and non-algorithmic models.

Algorithmic Models

Algorithmic models are designed in such a way that they provide a mathematical equation which is based upon the statistical analysis of data gathered from previously developed projects, e.g. Software Life Cycle Management (SLIM) [6] and COCOMO [7] and Albrecht's Function Point. These mathematical equations use inputs such as Source Lines of Code (SLOC), number of functions to perform / number of user screen, interfaces, complexity, and other cost drivers such as language, design methodology, skill-levels, risk assessments, etc. at a time when uncertainty is mostly present in the software [4,8]. As most of the software development effort estimates are based on the prediction of size of the system to be developed but this is a difficult task as the estimates obtained at the early stages of development are more likely to be inaccurate because not much information of the project to be developed is available at that time. So the correctness of algorithmic model largely depends upon the information that is available during the preliminary stages of development. Now considering the current technological advancements these algorithmic models are unable to provide a suitable solution. Though these models may be good enough to handle a particular environment but they are not flexible enough to adapt new environment.

The inability of algorithmic model to handle categorical data (which are specified by a range of values) and most importantly lack of reasoning capabilities contributed to the number of studies exploring non-algorithmic methods [17].

Non Algorithmic Models

Non-algorithmic techniques that came into existence in 1990's are based on new approaches such as, Parkinson, Expert Judgment, Price-to-Win and Machine Learning approaches. The soft computing techniques include methodologies like artificial neural networks, fuzzy logic, bayesian networks and evolutionary computing. Due to their inherent nature these techniques are used to handle imprecision and uncertainty [14]. These techniques handle real life vague situations by providing flexible information processing capability [16]. The principle of soft computing is to device methods of computation that leads to acceptable low cost solution, to an imprecisely/precisely stated problem [22].

Fuzzy Logic with its offerings of a powerful linguistic representation can represent imprecision in inputs and outputs, while providing a more expert knowledge based approach to model building. The first realization of the fuzziness of several aspects of COCOMO was carried out by Fei and Liu. They observed that an accurate estimate of delivered source instruction (KDSI) cannot be made before starting a project, and it is unreasonable to assign a determinate number for it [9]. Jack Ryder investigated the application of fuzzy modeling techniques to two of the most widely used models for effort prediction; COCOMO and the Function-Points models respectively [8]. Idri, Abran and Kjiri applied fuzzy logic to the cost drivers of intermediate COCOMO model [10]. MacDonell et al. developed a tool FULSOME that used fuzzy logic for data acquisition, model expression and knowledge gathering [23]. Musilek et al. presented the application of fuzzy logic to represent the mode and size as input to COCOMO model. They presented a two-stage implementation called simple F-COCOMO model and augmented F-COCOMO model. Ahmed et al. fuzzified the two parts of COCOMO model i.e., nominal effort estimation and the adjustment factor. They proposed a fuzzy logic framework for effort prediction by integrating the fuzzified nominal effort and the fuzzified effort multipliers of the intermediate COCOMO model [11]. Hodgkinson and Garratt represented that estimation by expert judgment was better than all regression based models. A marriage between neural networks and fuzzy logic, named Nero-fuzzy, was introduced in cost estimation. Nero-fuzzy systems can take the linguistic attributes of a fuzzy system and combine them with the learning and modeling attributes of a neural network to produce transparent, adaptive systems [2]. Venkatachalam investigated the application of artificial neural network to cost estimation [27]. Burgess et. al. applied genetic programming to carry software effort estimation [28].

Thus it can be summarized from the previous research that all soft computing based techniques lack in one aspect or the other and still there is lot of uncertainty in deciding that what soft computing based prediction technique should be applied to which prediction problem. In this paper a fuzzy logic based COCOMO II model is proposed to so as

to overcome the problem of imprecision and uncertainty.

III. COCOMO FRAMEWORK AND FUZZYLOGIC

The COCOMO 81 model is a regression based software cost estimation model. It was developed by Barry Boehm in 1981 and thought to be the most cited best known and the most plausible of all traditional cost prediction models [12]. COCOMO model can be used to calculate the amount of effort and the time schedule for software projects. This model comprises of three basic development modes: Organic (low complexity, small software team), Embedded (stringent constraints for ex. real time systems), and Semidetached (in between organic and embedded).

Though it was one of the stable models of its time but it had number of drawbacks [4] like it strictly gears toward traditional development life cycle model; i.e. custom software is build from precisely stated specifications and an assumption over here is that software requirements are already defined and stable; which is not always true.

It relies on LOC; and measuring LOC at very early stages of development leads to uncertainty and results in inaccurate estimation. Here success depends largely on using historical data which isn't always available.

It does not cope up with the current development environment like RAD and 4GL etc., thereafter COCOMO II was published that overcomes most of the drawbacks of COCOMO 81.

COCOMO II comprises of the following models [13] [24]:

Application Composition Model— This model assumes that systems are created from reusable components, scripting or database programming. This model involves prototyping efforts to resolve potential high-risk issues such as user interfaces, software/ system interaction, performance, or technology maturity. It is used during the early stages of development when prototype of user interface is available. Software size estimates are based on application points / object points, and a simple size/productivity formula is used to estimate the effort required. Object points include screens, user interface, reports, and components that are likely to be used.

Early Design Model – This model is used during early stages of the system design after the requirements have been established and is based upon incomplete data. It involves exploration of alternative software/system architectures and concepts of operation. At this stage, not enough is generally known to support fine-grain cost estimation. It makes use of function points or KSLOC and small number of cost drivers. Estimates are based on function points, which are then converted to number of lines of source code.

Reuse Model: This model is used to compute the effort required to integrate reusable components and/or program code that is automatically generated by design or program translation tools. It is usually used in conjunction with the post-architecture model.

Post Architecture Model: Once the system architecture has been designed, a more accurate estimate of the software size can be made. – It involves the actual development and maintenance of a software product. This model proceeds most cost-effectively if a software life-cycle architecture has been developed; validated with respect to the system's mission, concept of operation, and risk; and established as the framework for the product. It uses source instructions and / or function points for sizing, with modifiers for reuse; a set of 17 multiplicative cost drivers; and a set of 5 factors determining the project's scaling exponent. The model is given as:

$$Effort = A * (Size)^{B} * \prod_{i=1}^{17} E \qquad (1)$$

where, $B = 0.91 + 0.01 * \sum_{j=1}^{5} Scale\ Fact$

'A' is multiplicative constant and Size is the size of project measured in KSLOC/ Function Points/Object Points.

The Fuzzy Logic tool was introduced in 1965, by Lotfi Zadeh, and is a mathematical tool for dealing with uncertainty. It provides a technique to deal with imprecision and information granularity. The fuzzy theory provides a mechanism for representing linguistic constructs such as "many", "low", "medium," "often," "few." In general, the fuzzy logic provides an inference structure that enables appropriate human reasoning capabilities. On the contrary, the traditional binary set theory describes crisp events, events that either do or do not occur. It uses probability theory to explain if an event will occur, measuring the chance with which a given event is expected to occur [26]. It consists of four main components:

Fuzzifier- It converts the crisp input into a fuzzy set. Membership Functions are used to graphically describe a situation.

Fuzzy Rule Base- It uses if-then rules.

Fuzzy Inference Engine- A collection of if -then rules stored in fuzzy rule base is known as inference engine. It performs two operations i.e. aggregation and composition.

Defuzzification- It is the process that refers to the translation of fuzzy output into crisp output.

IV. PROPOSED RESEARCH METHODOLOGY

It is important to stress that uncertainty at the input level of COCOMO model results in uncertainty at output [15]. COCOMO II comprises of size, cost drivers and scale factors input and effort as output which is measured in person months (PM).The problem with software effort estimation is that it largely depends upon single values of size, cost drivers and scale factors. The size of the project is estimated based upon previously completed projects that are somewhat similar with the current project. Also cost drivers and scale factors need to have through assessment rather than assigning a fixed numeric value. To overcome this situation it would be better to represent these inputs in the form of fuzzy sets, in which interval values are, used which are represented by membership function.

In the proposed model COCOMO II's input parameters i.e. size, cost drivers and scale factors are taken into consideration.

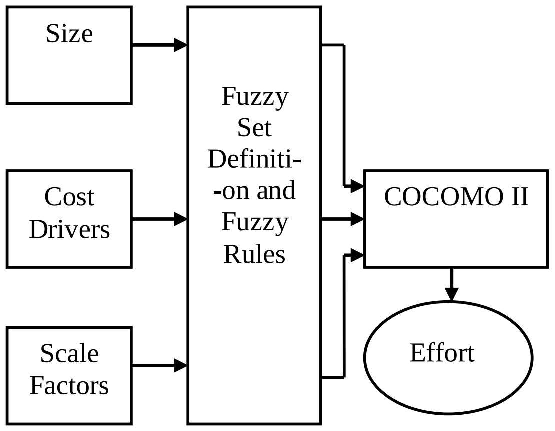

Figure 1. Fuzzy COCOMO Model

Gaussian Membership Function (GMF) is chosen as the results obtained by it are much better as compared to Triangular Membership Function (TMF) and Trapezoidal membership Function (Trapmf) [25]. They allow smoother transitions between the intervals.

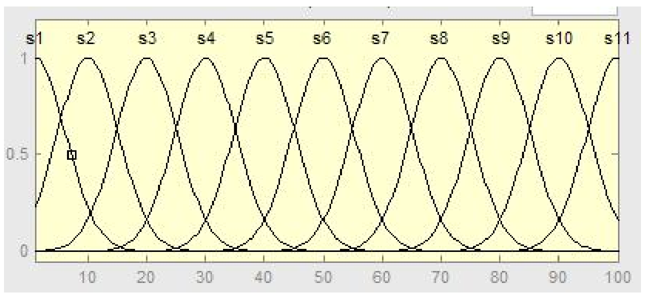

Figure 2.COCOMO input variable size represented as GMF

All these input variables are changed to fuzzy variables using fuzzy sets for each linguistic value such as very low, low, nominal, high, very high and extra high. as applicable to each cost driver and scale factor. For each cost driver a separate fuzzy inference system is designed. Rules are developed as cost driver in the antecedent part and corresponding effort multiplier in the consequent part. The defuzzified value for each of the effort multiplier is obtained from individual fizzy inference systems. Similarly scale factors are also fuzzified. The case of analyst capability (acap) cost driver is discussed as sample. Fuzzification of analyst capability is based upon COCOMO II 2000 Calibrated Post-Architecture model values (in Tables I and II) are shown in Fig. 3 and Fig. 4 as sample.

TABLE I
ACAP COST DRIVER RANGE DEFINED IN TERMS OF PERCENTILES

Very Low	Low	Nominal	High	Very High
15th	35	55	75	90

Table II
ACAP EFFORT MULTIPLIER RANGE DEFINED

Very Low	Low	Nominal	High	Very High
1.42	1.19	1	0.85	0.71

The proposed fuzzy based software effort estimation model rules contain linguistic variables related to the project. The rule base for fuzzy inference system (FIS) make use of connectives 'and/or' for COCOMO input variables to form number of rules. For fig. 3 and fig. 4 following rules are formed:

If (ACAP is Very Low) then (EFFORT is Increased Significantly)
If (ACAP is Low) then (EFFORT is Increased)
If (ACAP is Nominal) then (EFFORT is Unchanged)
If (ACAP is High) then (EFFORT is Decreased)
If (ACAP is Very High) then (EFFORT is Decreases Significantly)

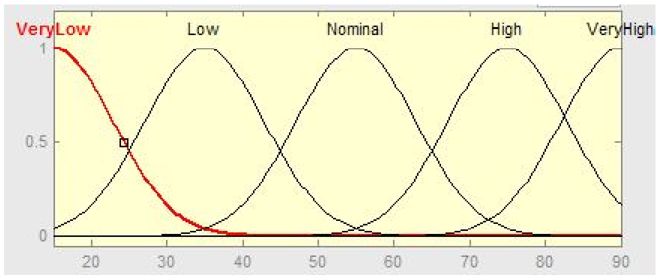

Figure 3. Antecedent of ACAP cost driver using Trapezoidal Membership Function

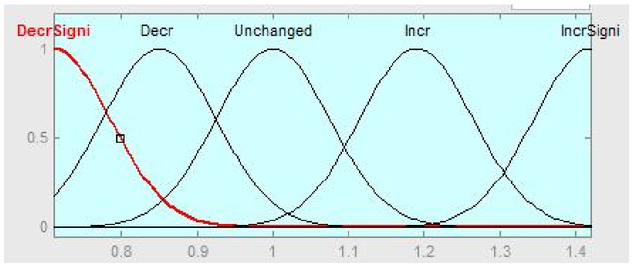

Figure 4. Consequent of ACAP cost driver using Trapezoidal Membership Function

V. EXPERIMENTAL RESULTS

The validation of the proposed model is carried out on a subset of projects from a repository of COCOMO dataset. A subset of dataset is applied to the proposed fuzzy model i.e. software development effort obtained using COCOMO II and efforts obtained by using fuzzy logic using gaussmf are calculated. The conversion of dataset of COCOMO 81 to COCOMO II is done using the Rosetta Stone [29]. It is observed that the effort obtained after applying fuzzy logic was closer to actual effort as compared to COCOMO II. The parameter used for evaluation of proposed model is MRE and is given by:

$$MRE = \frac{|\text{Actual Effort} - \text{Predicted Effort}|}{\text{Actual Effort}} \times 100 \quad (2)$$

MRE is calculated for COCOMO II as well as for the proposed fuzzy gaussian membership function. It is observed that MRE obtained for the proposed model is quite less as compared to MRE obtained by COCOMO II. Also it is observed that by increasing the number of fuzzy sets for input variable size to 5, 7, and 11 improves the performance further. The results are improved with 11 membership functions for size shown in Fig. 2. Fig. 5 shows the comparison of effort predicted by applying COCOMO II, FIS to that of actual effort. Fig. 6 shows the graphical representation of comparison of MRE of COCOMO II and FIS respectively.

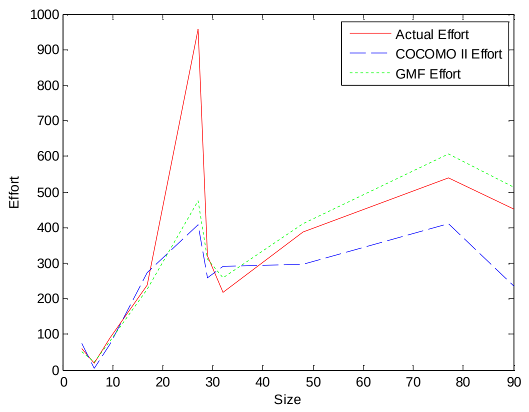

Figure 5. Comparison of Effort predicted by COCOMO II, FIS and Actual effort using COCOMO public dataset.

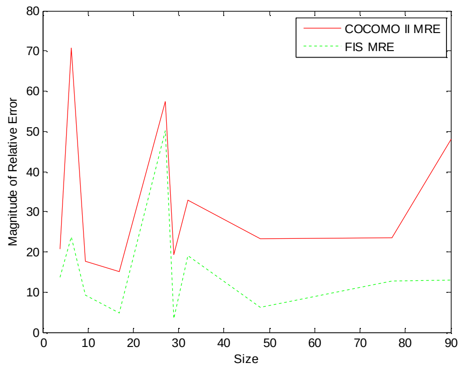

Figure 6. Comparison of MRE between COCOMO II model and proposed fuzzy model using COCOMO dataset.

The results show that the mean magnitude of relative error (MMRE) for COCOMO II is 32.6090 and for the proposed model it comes out is 15.5889. It shows that the proposed model has lesser MMRE than COCOMO. Also in case of Pred the proposed model show that pred (20) is 80% compared to that of COCOMO II's 30%, thus showing better accuracy.

VI. CONCLUSION AND FUTURE SCOPE

The study reveals that the proposed fuzzy logic based COCOMO II model overcomes the uncertainty in the inputs that is present in the traditional COCOMO and thus improves the accuracy of software effort estimation. By determining more suitable fuzzy rule sets and by deploying technologies like type-2 fuzzy uncertainty can be handled more closely and thus more accurate software effort estimation is possible.

REFERENCES

[1] S.G. MacDonell, and A. R. Gray, "A comparison of techniques for software development effort prediction", *International Conference on Neural Information Processing and Intelligent Control Systems, New Zealand*, pp: 869-872, 1997.

[2] A.C. Hodgkinson, and P.W. Garratt," A neurofuzzy cost estimator", *Proceedings of Third International Conference on Software Engineering and Applications*, pp: 401-406, 1999.

[3] Mockus A., Weiss D.M. and Zhang P., "Understanding and Predicting Efforts in Software Projects", *IEEE Proceedings of 25th International Conference on Software Engineering (ICSE'03)*, pp. 274-84.

[4] Wu L., "The Comparison of Software cost estimation methods", University of Calgary.

[5] I. Somerville, *Software Engineering*, 6th ed., Addison–Wesley Publishers Limited, 2001.

[6] L.H. Putnam, "A general empirical solution to the macro software sizing and estimating problem", *IEEE transactions on Software Engineering*, vol. 2, pp: 345-361.

[7] B. W. Boehm, *Software Engineering Economics*, Englewoods Cliffs, NJ, Prentice-Hall, 1981.

[8] J. Ryder, "Fuzzy modeling of software effort prediction" *IEEE Information Technology Conference*, pp: 53-56, 1998.

[9] Z. Fei and X. Liu., "f-COCOMO: fuzzy constructive cost model in software engineering", *IEEE International Conference on Fuzzy Systems*, pp: 331-337, 1992.

[10] A. Idri, A. Abrian, and L. Kjiri, "COCOMO Cost Model using Fuzzy Logic", *International Conference on Fuzzy Theory & Technology Atlantic, New Jersey*, 2000.

[11] P. Musilek, W. Pedrycz, G. Succi, and M. Reformat," Software cost estimation with fuzzy models", *Applied Computing Review*, vol. 2, pp: 24-29, 2000

[12] C.L. Martin, J. L.Pasquier, Y. M. Cornelio and G. T. Augustin, "Software development effort estimation using fuzzy logic: A case study" *IEEE Proceedings of the Sixth Mexican International Conference on Computer Science (ENC)*, pp: 113-120, 2005.

[13] B. Boehm, B. Clark, E. Horwitz, R. Madachy, C. Abts, S. Chulani, A. W. Brown and B. Steece, "COCOMO II model definition manual", *University of South California Center for Software Engineering*, 2000.

[14] M.W. Nisar, J. W. Yong and M. Elahi, "Software development effort estimation using fuzzy logic - A survey", *IEEE International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 1, pp: 421-427, 2008.

[15] M. Jorgenson and D.I.K. Sjoberg, “The impact of customer expectation on software development effort estimates”, *International Journal of Project Management*, vol. 22, pp: 317-325

[16] H. K. Verma, and V. Sharma, “Handling imprecision in inputs using fuzzy logic to predict effort in software development”, *IEEE International Advance Computing Conference (IACC)*, pp: 436-442, 2010.

[17] I. Attarzadeh, and S.H. Ow, “A novel soft computing model to increase the accuracy of software development cost estimation”, *IEEE International Conference on Computer and Automation Engineering (ICCAE)*, vol. 3, pp: 603-607, 2010.

[18] M.O. Saliu, M.A. Ahmed, and J. AlGhamdi, “Towards adaptive soft computing based software effort prediction”, *IEEE Annual Meeting of Fuzzy Information*, vol. 1, pp: 16-21,2004.

[19] J. M. Mendel, “Fuzzy logic system for engineering: A tutorial”, *IEEE Transactions on Neural Networks*, vol. 11, pp: 748-768, 2000.

[20] K. Strike, K. E. Emam, and N. Madhavji, “Software cost estimation with incomplete data” *IEEE Transactions on Software Engineering*, vol. 27, 2001.

[21] Srinivasan, K. and Fisher D., “Machine Learning Approaches to Estimating Software Development Effort”, *IEEE Transactions on Software Engineering*, Vol.21, No. 2, 1995.

[22] Mitra, S. and Hayashi, Y.: “Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework”, *IEEE Transaction on Neural Networks*, Vol. 11, No. 3, May 2000.

[23] MacDonell, S.G. Gray, A. R. and Calvert, M.J., “FULSOME: A Fuzzy Logic for Software Metric Practitioners and Researchers”. *IEEE Proceedings of the 6th International Conference on Neural Information Processing (ICONIP ’96)*, pp. 308-313.

[24] Boehm B, Clark B., Horwitz E., Madachy R., Westland C. and Selby R., “The COCOMO 2.0 software cost estimation model”.

[25] Kad S. and Chopra V., “Software Development Effort Estimation using Soft Computing”, *IEEE Proceedings of 3rd International Conference on Machine Learning and Computing (ICMLC ’11)*, Vol. 5, pp. 205-208, 2011.

[26] Sivanandam S. N., Sumathi S. and Deepa S.N., *Introduction to Fuzzy Logic using MATLAB*, Springer- Verlag, 2007

[27] Venkatachalam, A.R., "Software cost estimation using artificial neural networks", *IEEE Proceedings of International joint Conference on Neural Networks(IJCNN '93)*, pp: 987-990.

[28] Burgess C.J. and Lefley M., "Can genetic programming improve software effort estimation? A comparative evaluation", *Information and Software Technology*, Vol. 43, Issue 14, pp: 863 -873.

[29] Reifer D.J., Boehm B.W. and Chulani S., "The Rosetta Stone: Making COCOMO 81 files work with COCOMO II", University of South California.