Soft computing based technique for accurate effort estimation: A survey

Sangeeta Bhandari[1],Parveen Kakkar[2]
[1]HMV, Jalandhar,[2]DAVIET,Jalandhar
[1]sangeetakapoorbhandari@gmail.com, [2]parveenkakkar@rediffmail.com

Abstract

The global software market has grown exponentially over the past decade. The cost of developing software has grown significantly. The main reason attributed to this increasing trend in the software costs is the labor intensive nature of the software development process. To effectively manage software projects it is important to have accurate estimates of cost and effort involved in software development. The number of project failures and the cases of cost and schedule overrun have been a significant issue for software project managers. Poor estimates have not only led projects to exceed budget and go overscheduled but also in many cases to be terminated entirely. Software cost estimation is the set of techniques and procedures that organizations use to arrive at an estimate for proposal bidding, project planning and probability estimates. Accurate estimate means better planning and efficient use of project resources such as cost, duration and effort requirements for software projects. Efficient software development effort estimation is one of the most demanding tasks in software industry. Unfortunately software industry suffers from the problem of inaccurate estimate for projects and in many cases inability to set correct release date, leads to low quality of delivered product. In this paper we have explored and analyzed different soft computing based techniques employed for software effort estimation based on COCOMO model(Boehm B.).

Keywords : COCOMO, Soft computing, Fuzzy logic, Software cost estimate, Neural network

1. Introduction

Software effort estimation is the process of predicting the amount of time (Effort) required to build a software system. In order to perform cost-benefit analysis, cost estimation is to be performed by client or developer. Cost Estimation is achieved in terms of person-months (PM), which can be translated into actual dollar cost. Estimation carries inherent risk and this risk leads to obscurity. The different obscurity factors are project complexity, project size etc. The concept of software cost estimation has been growing rapidly due to practicality and demand for it. Today the people are expecting high quality software with a low cost, the main objective of software engineering. So many popular cost estimation models like COCOMO81, COCOMOII, SLIM, FP, Delphi, Halsted Equation, Bailey- Basili, Doty, and Anish Mittal Model had came into existence. These models are created as a result of regression analysis and power regression analysis methods applied to historical data. In spite of availability of software cost estimation models accurate estimation of software development cost continues to challenge software engineering researchers due to the continued lack of accurate estimates. Software industry suffers from the problem of inaccurate estimate for projects and in many cases inability to set correct release date, leads to low quality of delivered product. In this paper we have explored and analyzed different soft computing based techniques employed for software effort estimation based on COCOMO model (Boehm B.).

2. Software Cost Estimation Process:

In the early days of computing software costs constituted a small percentage of the overall computer-based system cost. An error in estimates of software cost had relatively little impact. Today software is most expensive element of all computer based systems. Historically a decomposition technique that takes a divide and conquers approach to software project estimation has been used. These techniques decompose the project into major functional units and related software development activities. The cost and effort estimation of these subunits is performed in a stepwise fashion. The models proposed for software cost estimation are based on historical data. These are of the form

$$\mathbf{D=f(v_i)}$$

Where D is parameter to be estimated (effort, cost, duration etc.) and $v_{i\ are}$ independent parameter like LOC or function points.

Now decomposition techniques may make estimate either by dividing problem into sub problems i.e. problem based estimation or may consider steps of software development process i.e. process based estimation.

In case of problem based estimation LOC and function point data are used during software project estimation. The LOC and function point estimation technique differ in the level of detail required for decomposition. When LOC is used as the estimation variable, decomposition is essential to a considerable level of detail. Each and every function to be implemented is to be identified and studied in detail. For function point estimates, rather than focusing on functions, each of the information domain characteristic i.e. inputs, outputs, data files, external interfaces and complexity adjustment values are estimated. The resultant estimates can then be used to derive a FP value that can be extracted from past data and used to generate an estimate. Using historical data the planner estimates an optimistic, most likely and pessimistic size value for each function or count for each information domain value. The expected value for estimation variable S can then be computed as a weighted average of optimistic, most likely and pessimistic values i.e.

$$S=(S_{OPT} + 4_{SM} + S_{PESS})/6$$

In process based estimation, the planner estimates the effort (e.g. person months) required to accomplish each software development process activity e.g. analysis, design, coding etc. From above discussed two approaches the problem based estimation has normally been used. These two approaches give us an initial approximation for LOC or FPs. An estimation model for software uses empirically derived formula to predict effort as a function of LOC or FP. The empirical data that supports most estimation models are derived from limited sample of projects. Therefore no estimation model is appropriate for all classes of software and in all development environments. This is basic reason for inaccurate cost estimation. Various LOC-oriented estimation model found in literature are:

Boehm Simple model	$E=3.2 * (KDLOC)^{1.05}$
Bailey-Basli model	$E=5.5 + 0.73 * (KDLOC)^{1.16}$
Walston-Felix model	$E=5.2 * (KDLOC)^{0.91}$
Doty model for KLOC>9	$E=5.288 * (KDLOC)^{1.047}$

Various FP oriented models have also been proposed like

Kemerer model	$E=60.62 * 7.728 \text{ x } FP^3$
Matson, Barnett and Mellichamp	$E=585.7 + 15.12FP$

The general form of these model may be represented by

$$\mathbf{E=A+B * (ev)^C}$$

As we may observe from these models, each gives different result for same value of delivered lines of code or for same number of function points. So these models must be adjusted for a particular context before application.

Apart from above mentioned models COCOMO i.e. Constructive Cost Model has been most widely applied and studied. Boehm described COCOMO as a collection of three variants: basic model, intermediate model and detailed model. The basic COCOMO model computes effort as function of program size, and it is same as single variable method.

Effort =a * size^b

An intermediate COCOMO model effort is calculated using a function of program size and set of cost drivers or effort multipliers also called effort adjustment factors.

Effort = (a * size^b)EAF

In detailed COCOMO the effort is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle. The phases used in detailed COCOMO are requirements planning and product design, detailed design, code and unit test, and integration testing.
Effort = (a * size^b)*EAF * sum(Wi).

for embedded systems: a=3.6, b=1.20
for organic systems: a=2.4, b=1.05
for semi-detached systems: a=3.0, b=1.12.

3. COCOMO II Model

Boehm and his colleagues have refined and updated COCOMO called as COCOMO II. This consists of application composition model, early design model, post architecture model.

1) The Early Design Model

It uses to evaluate alternative software system architectures where unadjusted function point is used for sizing.
Effort = a*KLOC*EAF

2) The Post Architecture Model

It is used during the actual development and maintenance of a product. The post architecture model includes a set of 17 cost drivers and a set of 5 factors determining the
projects scaling component.
Effort=(a*size ^{b})*EAF
Where a=2.55 and b is calculated as b=1.01+0.01*SUM(Wi)
wi= sum of weighted factors.

Now in spite of years of research and improvements in the models used for software cost estimation, software estimation is deemed to be a tough nut to crack. Still there are cases of schedule and cost overruns. Moreover today software size has become enormous, so has become the importance of cost estimation.

From last three to four years new techniques like artificial neural networks, genetic algorithms, and fuzzy logic is being employed to deal with the uncertainties in the inputs to the models. Because of inherent uncertainties in the inputs, cost estimation has suffered. The use of soft computing techniques has been demonstrated by many researchers, which is the main idea of this paper.

The basic point of contention regarding above mentioned class of models is concerned with the knowledge of precise (and numeric) values of the size of code prior to the completion of the project itself. It is very likely that our knowledge regarding the anticipated size of the system, especially at an early stage of the project, will not be detailed and precise. Any numeric estimate could be somewhat illusionary.

Venkatachalam A.R.(1993) was of opinion that although algorithmic models provide an economical approach to estimate software costs, they suffer from some serious weaknesses. First, the cost and effort estimates derived from different models seem to have significant variations (Saiedian, Band, and Barney, 1992). Such large variations may pose problems to managers in deciding the amount of resources to be committed. Second, the models are based on historical data and hence may not reflect recent developments in the areas of programming languages, hardwares, and software engineering. So he has proposed the use of artificial neural networks for accurate cost estimation.
According to Venkatachalam back propagation neural network is most appropriate to be used in this context. A back-prop neural network is organized in layers, with each layer composed of neurons or processing elements and connections (Rummelhart et al. 1986). The first layer called the input layer contains neurons that represent the set of input variables. The output layer contains neurons that represent the output variables. When the relationship between the input and output variables is nonlinear, the hidden layer helps in extracting higher level features and facilitate generalization. Connections between neurons have numerical weights associated with them; the weights are adjusted in the training process by repeatedly feeding examples from the training set. Each neuron has an activation level, specified by continuous or discrete values. The value (called internal activation) coming into a neuron in the hidden or output layers is typically the sum of each incoming activation level times its respective connection weight.In back-prop network paradigm, all the connection weights are assumed to be responsible for the output error. Error is defined to be the difference between a network's estimated output or predicted value and the corresponding observed output value. The error values are calculated at the output layer and propagated to previous layers and used for adjusting the connection weights. The training process consists of repeatedly feeding input and output data from empirical observations, propagating the error values, and adjusting the connection weights until the error values fall below a user-specified tolerance level. In his work Venkatchalam A.R. has used a back-prop neural network is constructed with 22 input nodes and 2 output nodes. The input nodes represent the distinguishing features of software projects and the output nodes represent the effort required in terms of person month and the development time required to complete the project. The projects considered for this research are taken from the COCOMO database (Boehm, 1981). The input nodes represent the various factors affecting the software cost i.e. Type of project (business, process control, scientific, etc.), Programming language used in the project, Required software reliability, Database size, Product complexity etc.

According to W. Pedrycz et.al.in 1999, It is more appropriate to view an estimate of the size of code as a fuzzy set. They have used triangular fuzzy number, viz. a fuzzy set defined in **R** with a piecewise linear membership function**.** A triangular fuzzy number is described by three parameters –its modal value (m) and two bounds (lower and upper, “a” and “b”, respectively) beyond which value of the membership function is assumed to be zero. The lower bound expresses a minimum possible size of the system whereas the upper bound stands for the maximal anticipated size of the software system. The calculus of fuzzy numbers has been used to calculate effort for the given triangular fuzzy set. This representation makes it possible to envision each data item in form of multiple possibilities rather than discrete values.
Another problem with the existing cost estimation models which are regression models is that, in spite of their nonlinear nature are global. They attempt to capture all data within a single relationship (function). The idea behind granular models and granular computing is that data are captured through a series of local models being constructed on the basis of individual information granules. This paper demonstrates the use of fuzzy clustering to deal with heterogeneous data that naturally arise due to the nature of the problem (promoting a substantial diversity of the software projects) the models produce a granular form of the results that tend to be more informative and comprehensive than single numeric. It has been found that the concept of information granularity and fuzzy sets, in
particular, plays an important role in making these models more user-friendly.

Idri A.,Khoshgotaar T.M, Abran A.(2002) state that main reasons for skepticism about neural network is their nature of being black boxes. It implies that it is difficult to explain the reason about particular output given by neural network. This is a significant weakness as without the ability to produce comprehensible decisions, it is not possible to trust the reliability of neural networks that deal with real world problems. Idri A.,Khoshgotaar T.M, Abran A. have proposed the use of methods that maps the neural network to a fuzzy rule based system. So if these rules are comprehensible then the neural network may also be easily interpreted. They have considered a three layer perceptron neural network with the sigmoid function for the hidden units and identity function for output unit. After training and testing the network with COCOMO'81 dataset, they have applied the Benitez's method( ) to extract the if-then fuzzy rules from the network. These rules express the information encoded in the architecture of the network.

Attarzadeh I.,Hock Ow S.(2010) has also proposed a new cost estimation model based on artificial neural networks. The proposed structure of network is customized for COCOMO-II Post-architectural model. The proposed model was evaluated both by using original data from COCOMO dataset and artificial dataset. The results have shown that proposed neural network model showed better software effort estimates

Fuzzy logic has also been used to remove vagueness in inputs to cost estimation problem. Swarup Kumar J.N.V.R. et.al.(2011) has used the concepts of fuzzification and defuzzification for handling ambiguity and impreciseness in the inputs to COCOMOII. They have compared the values of effort calculated by eight models, which included COCOMO Basic Model, COCOMO Inter (Nom), Detailed (Nom), Early Design Model (High), post Arch Model (H - H), Doty Model, Mittal Model and Swarup Model. The results demonstrate that applying fuzzy logic method to the software effort estimation is an expedient approach to address the problem of obscurity and vagueness existing in software effort drivers. Also, the fuzzy logic model presents better estimation accuracy as compared to the other models. The performance of proposed software effort estimation
model has been evaluated by comparing against various software cost estimation models. All these models use Mean Relative Error (MRE) as evaluation criteria.
For each model, the impact of estimation accuracy was evaluated using (MRE, MARE) evaluation criteria. Criterion for measurement of software effort estimation model performance is Mean Absolute Relative Error (MARE).

Jin-Cherng Lin and Han-Yuan Tzeng(2010) have applied Particle Swarm Optimization to estimate software effort by multiple factors software project clustering.

While using PSO to optimize the parameters of COCOMO, each particle contains two dimensions X and Y coordinates. X and Y are A and B parameters in the COCOMO model equation respectively. PSO helps in finding optimal values of A and B parameters as the prediction project parameters. First, X and Y coordinates are randomly generated 40 particles of range between 0 and 1 in a two-dimensional space, and each particle has random initial speed. Then the X and Y coordinates of particles act as predictor parameters, and use MMRE as fitness Value. Each particle must acquire an optimal value on their path; this solution is called the local optimal solution (Pbest). Each particle must also have social behavior so that each particle finds the optimal solution in the current search, which is called a global optimal solution (Gbest).
Although this work was first in applying correlation and intelligent computing for software cost estimate, but it was difficult to implement and no concrete results were reported.

J.N.V.R.Swarup Kumar et.al.(2011) have proposed a new model using fuzzy logic in order to estimate software effort. They have used MATLAB to determine the parameters of various cost estimation models.

Attarzadeh I.,Hock Ow S.(2011) have used Adaptive fuzzy logic model to improve the accuracy of software time and cost estimation. They have used two-dimension Gaussian Membership Function in

fuzzy model to make software attributes smoother in terms of the range of values. The proposed model was applied on COCOMO I and NASA98 datasets. The evaluation of obtained results using mean of magnitude of relative error showed the superiority of fl-COCOMO model over original COCOMO.

4. Conclusion

Most important issue in software project management is accurate and reliable estimation of software development effort and cost. This is more important especially in the early phase of software development so that the manager may commit his resources for on time delivery of the software. Software attributes like LOC and function points on which traditional models of cost estimation are based have properties of uncertainty and vagueness. A software cost estimation model based on soft computing techniques can overcome the uncertainty and vagueness of software attributes. However as shown by various researches carried out, determination of suitable fuzzy rule sets for fuzzy inference system and suitable architecture in case of neural network plays important role in coming up with accurate and reliable software cost estimates. Accuracy of the models proposed is still an issue but direction of application of soft computing techniques in software cost estimation seems promising and must be carried further.

References:

1. Ridhika.S. (2013), *Survey: Non-Algorithmic models for estimating software effort*. European International Journal of Science and Technology ISSN: 2304-9693.
2. Iman Attarzadeh, Siew Hock Ow (2011), *Improving Estimation Accuracy of the COCOMO II Using an Adaptive Fuzzy Logic Model.* IEEE International Conference on Fuzzy Systems June 27-30, Taipei, Taiwan
3. Mohd. Sadiq, Farhana Mariyam, Aleem Ali1, Shadab Khan, Pradeep Tripathi (2011). *Prediction of Software Project Effort Using Fuzzy Logic*. IEEE .
4. Swarup J.N.V.R.Kumar et.al(2011). *A Novel Method for Software Effort Estimation Using Inverse Regression as firing.* 978-1-4244-8679-3/11, IEEE.
5. Verma H.K and Sharma V. (2010). *Handling Imprecision's in Inputs using Fuzzy Logic to predict effort in Software Development***.** Proceedings of IEEE International Advance Computing Conference, ISBN: 978-1-4244-4790-9, pp: 436-442.
6. Junhai M. and Lingling M. (2010), *Comparison Study on Methods of Software Cost Estimation.* IEEE 2nd International Conference on e-Business and Information System Security (EBISS), pp: 1-4.
7. Jin-Cherng Lin, Han-Yuan Tzeng(2010), Dept. of Computer Science & Engineering, Tatung University,Taipei, Taiwan, *Applying Particle Swarm Optimization to Estimate Software Effort by Multiple Factors Software Project Clustering*, IEEE.
8. Hamdan K.,Khatib H.E. and Shuaib K., (2010) *Practical software project total cost estimation method."* IEEE International Conference on Multimedia Computing and Information Technology (MCIT)", ISBN: 978-1-4244-7001-3, pp: 5-8.
9. Attarzadeh I. and Ow S.H.,(2010). *A Novel Soft Computing Model to increase the accuracy of Software Development Cost Estimation*, Proceedings of IEEE International Conference on Computer and Automation Engineering, ISBN: 978-1-4244-5585-0, Vol. 3, pp: 603-607.
10. Mittal Anish, Parkash Kamal, Mittal Harish (2010), *Software Cost Estimation using fuzzy logic*, ACM SIGSOFT Software Engineering Notes, Volume 35.
11. Mittal H., Bhatia P.(2009), *Software Maintainability Assessment based on fuzzy logic Technique* ACM SIGSOFT Vol 34 No 3.
12. Hari CH.V.M.K. et. al (2009), *Identifying the Importance of Software Reuse in COCOMO81, COCOMOII*. International Journal on Computer Science and Engineering Vol.1 (3),142-147, ISSN: 0975-3397.
13. Sheta, D. Rine and A. Ayesh (2008), *Development of Software Effort and Schedule estimation Models Using Soft Computing Techniques* ,IEEE Congress on Evolutionary Computation(CEC),.
14. Galindo J. (2008) *Handbook of Research in Fuzzy Information Processing in Databases*, Information science Reference.
15. Nisar M.W., Yong J. W. and Elahi, M.(2008), *Software Development Effort Estimation Using Fuzzy Logic - A Survey***,** IEEE Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Vol. 1, pp: 421-427.
16. Sun-Jen Huang , Nan-Hsing Chiu (2007). *Applying fuzzy neural network to estimate software development effort*. Springer Science+Business Media, LLC,pp.73-83.
17. Magne Jorgensen, Stein Grimstad (2005), Simula Research Laboratory,Norway, *Over-Optimism in Software Development Projects: TheWinner's Curse*, (CONIELECOMP-2005).
18. Menzies T., Port D., Chen Z., Hihn J., and Stukes S. (2005), *Validation Methods for calibrating software effort models*, in ICSE '05: Proceedings of the 27th international conference on Software engineering, (New York, NY, USA), pp. 587–595, ACM Press.
19. Shan, Y. ,McKay, R.I. ,Lokan, C.J. ,Essam, D.L. (2004) *Software Project Effort Estimation Using Genetic Programming.* IEEE Communications.
20. Jalote Pankaj,2004 *An Integrated Approach for Software Engineering*, Third Edition. ISBN: 978-81-7319-702-4.

21. K. Molokken and M. Jorgensen, (2003). *A review of software surveys on software effort estimation,* Proceedings of IEEE International Symposium on Empirical Software Engineering (ISESE 2003). Rome, Italy, pp: 223 – 230.
22. Zonglian F. (2003), *f-COCOMO: Fuzzy Constructive Cost Model in Software Engineering***,** IEEE International Conference on Fuzzy Systems, ISBN: 0-7803-0236-2, pp: 331-37.
23. Strike K., Emam K.E., and Madhavji N. , (2001), *Software cost estimation with incomplete data***,** IEEE Transactions on Software Engineering, Vol. 27, Issue: 10.
24. Somerville I. (2001), *Software Engineering***,** Sixth Edition, Addison –Wesley Publishers Limited,.
25. Lofti Zadeh (2001), *The future of soft computing***,** The 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, pp. 217-228.
26. Hale, J., Parrish A., Dixon B. and Smith R.K. (2000), *Enhancing the COCOMO estimation models*, IEEE Software Journal, Vol. 17, Issue 6, and pp: 45-49.
27. Boehm B. (2000), *Safe and simple software cost analysis*, IEEE Software Journal, Vol.5, Issue: 17, pp: 14-17.
28. Ali Idri and Alain Abran ,(2000), *A Fuzzy Logic Based Set Of Measures For Software Project Similarity: Validation And Possible Improvements***,** Sixth Maghrebian Confereence on Computer Sciences, pp. 9-18.
29. Pedrycz W. ,.Peters J.F, Ramanna S. (1999*), A Fuzzy Set Approach to Cost Estimation of Software Projects*, Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering, Shaw Conference Center, Edmonton, Alberta, Canada.
30. J. Ryder, (1998). *Fuzzy modeling of software effort prediction.* Proceedings of IEEE Information Technology Conference, Syracuse, NY, pp 102-114.
31. Shi, Y. Eberhart, R. (1998) *A modified particle swarm optimizer*. IEEE Evolutionary Computation Proceedings .
32. Pedrycz W., Gomide F. (1998), *An Introduction to Fuzzy Sets: Analysis and Design*, MIT Press, Cambridge, MA,.
33. Chrysostomos D. Stylios , Voula C. Georgopoulos , Peter P. Groumpos, (1997), *The Use of Fuzzy Cognitive Maps in Modeling Systems.* Proceeding of 5th IEEE Mediterranean Conference on Control and Systems, Paphos.
34. Benitez J M ,Castro J L, Requena I(1997), *Are Artificial Neural Networks Black Boxes?*, IEEE Transaction on Neural Networks, Vol. 8, No. pp-1156-1164
35. Boehm B.(1995) *Cost Models for Future Life Cycle Process: COCOMOII*.Annals of Software Engineering.
36. Kennedy, J. and Eberhart, R. C.(1995). Particle *swarm optimization. Proc. IEEE Int'l. Conf. on Neural Networks*, IV, 1942–1948. Piscataway, NJ: IEEE Service Center .
37. Lotfi Zadeh, A.,(1994).*Fuzzy Logic, Neural Networks and Soft Computing*, Communication of ACM., 37(3): 77-84.
38. Venkatachalam A.R., (1993). *Software Cost Estimation Using Artificial Neural Networks*, Proceedings of International Joint Conference on Neural Networks.
39. Fei Z. and Liu X. (1992), f-*COCOMO: Fuzzy Constructive Cost Model in Software Engineering***,** IEEE International Conference on Fuzzy Systems, ISBN: 0-7803-0236-2, pp: 331-337.
40. Boehm B (1981), *Software Engineering Economics Englewood Cliffs*, NJ,Prentice Hall.
41. Robert W. Zmud, Chris F. Kemerer, (1987) *.An Empirical Validation of Software Cost Estimation Models* Communication of the ACM Vol 30 No 5.
42. Baiely J., Basili W (1981), *A Metamodel for Software Development Resource Expenditure*. Proc. Intl. Conference Software Egg. pp : 107-115.