An Early Stage Effort Estimation from Enhanced use Case Point Analysis using Adaptive Neuro Fuzzy Inference System

Ch Prasada Rao
Assoc. Professor, CSE
Aditya Engineering College
Andhra Pradesh, India

Dr. Sripada Rama Sree
Professor, CSE
Aditya Engineering College
Andhra Pradesh, India

Abstract: **In the software development life cycle (SDLC), communication and planning phases are very crucial to develop the software product successfully. In Communication phase, requirements elicitation is done and in planning phase, effort estimation is done. Software project is termed to be successful if it is delivered to the customer within schedule, within budget, and with high quality. The previous reports clearly show that around 60-70% of the projects have failed due to over budget and over schedule. Software effort estimation in the early stage is very essential for the software managers to go for a better business case. In the business case, there exists a contract between the customer and manager based on cost, functionality, delivery deadline and quality constraints. One of the extreme approaches which have been widely used in the past two decades for the prediction of size, effort and cost is Use Case Points (UCP). Use case points are calculated from the Use Case diagrams to estimate the software effort in the early stages of SDLC. An improved version of UCP available in the literature is Enhanced Use Case Points. In this paper, a novel method of predicting the software effort from the Enhanced Use Case Points using Adaptive Neuro-Fuzzy Inference System (ANFIS) is proposed. The anticipated model was assessed based on VAF, MMRE, MMER, MSE and PRED using two datasets available. Software Development Effort estimated using the proposed approach is compared with the Karner's Method. For all performance measures, the proposed approach produced more accurate results.**

Keywords: SDLC, ANFIS, EUCP, UCP, Software Effort Estimation, Soft Computing

I. INTRODUCTION

As Pressman stated in his book [23], Software Engineering is "the Establishment and use of sound engineering principles in order to obtain economically software that is a reliable and works efficiently on real machines". From this, it can be inferred that, the accomplishment of software project relies on upon the way the tools and techniques are utilized for the improvement. The most prominent phases in the SDLC are Communication and Planning. In Communication phase, the requirements are discovered from the stakeholders using different techniques like questionnaire based, brainstorming, prototyping, Interviews, observations etc. In planning phase, the estimations of size, effort, and cost are predicted. The schedules and delivery deadlines are also determined. Accurate estimation of the software product advancement exertion in the early phase of the SDLC leads to success of the software project.

Software development Effort Estimation is an essential activity in software development process and software project management. Inaccurate software development effort assessment leads to over budget, over schedule and thereby may be a reason for failure of project. As indicated by the Standish chaos Report, 65% of software projects are conveyed over spending plan and after conveyance date [21]. As indicated by International Society of Parametric Analysis (ISPA) [3] and Standish Group International [4], the main reason behind the failure of software projects is improper estimation of software effort due to unclear, uncertain, ambiguous and incomplete software functional requirements. Improper software effort estimation also influences the Return on Investments (ROI) of the organization.

The literature shows the development of several software effort estimation techniques within the last three decades. These estimation techniques are grouped under three categories [8]:

1. *Algorithmic Models*: these are conventional models but still are more popular [10]. These models includes COCOMO [11], SLIM [12], SEER-SEM [13], ESTIMACS [17], Jansen Model [17], FPA [17], Soft Cost [17] etc . The main cost drivers of the model are software size, eventually the SLOC or KLOC and functional points. In Algorithmic models, cost is analyzed using mathematical equations. The equations are derived come from the analysis of past data [17] .
2. *Expert Judgment*: In this technique, Senior Software effort estimators use their experience and domain knowledge is based on past data and related estimated project data to forecast the software development effort. This technique is subjective, précised and it lacks proper standards. This approach is not reusable. The main drawback of this model is used phrases frequently like 'I believe that' ….or ' I feel it' [9]
3. *Machine learning models*: These are current and efficient techniques, alternate to algorithmic

models for effort estimation. Machine learning includes neural network, fuzzy algorithms, neuro-fuzzy logic, and genetic algorithms, regression trees etc.

The prediction of Software development effort is essential in the earlier phase of life cycle to project managers for staffing and organizing the activities successfully. Conventionally, we used SLOC or KLOC and FPA to estimate the cost of the project [23]. SLOC or KLOC is not suitable in the early phase of SDLC. SLOC or KLOC are programming language dependent and these will be collected only after completion of the Coding [23]. FPA is one of the most punctual models to gauge the size, expense and exertion of the software product in the early stage [23]. The FPA model was proposed by Albrecht in 1979 [5] and it ascertain the size of the product by its functionalities. The main advantage of FPA is technology independent and the drawback of the FPA is that the functional points cannot be computed automatically [6]. The only way to estimate software effort accurately in early stage by considering the functional & non functional requirements. Functional requirements are represented using the Use Case diagrams in UML and which are available in the Software Requirement Specification (SRS) Document.

Non-Functional requirements can be assed using the 13 technical factors and 8 Environmental parameters. Use case diagrams play a vital role in the Estimation of the Software Effort by using the Use Case Point (UCP) model [14]. UCP model was proposed by Gautsav Karner in 1993[7]. Now a days, there has been a lot of evolution in predict the software development effort using Machine Learning techniques. In this paper, an Adaptive Neuro Fuzzy Inference System (ANFIS) is used to Estimate the Software Effort in early stage of SDLC in terms of man-hours using EUCP method. ANFIS is the Sugeno-based Fuzzy Inference system that combines the characteristics and principles of Neural Network and Fuzzy logic into a single framework [17]. The present model is experimented with two different datasets which are openly available [15] [16]. The results of proposed ANFIS method are compared with the conventional Karner's method. It has been observed that, the ANFIS method produced better results in prediction of software development effort.

II. BACK GROUND AND RELATED WORK

Enhanced UCP Approach: Enhanced Use Case point (EUCP) method [15] increases the scope of the existing UCP method proposed by Karner in 1993. For estimating effort EUCP method follows the following steps.

A) Classification Of Actors and Use Cases:

The Actor is characterized into five classifications and allotted weights as needs be [15].

Unadjusted Actor Weights (UAW):

1) Simple Actor: If the actor is simple then he represents the system with the Application Program Interface (API). The corresponding weight for the Simple Actor is 1.
2) Average Actor: An Actor that communicates either Use case or System through some protocols like HTTP, FTP, other user defined etc, and then the Actor is called as Average. The corresponding weight for Average Actor is 2.
3) Average complex Actor: These Actors can be either system concerns or human related which can be supported only one transactions or corresponding weight is 3.
4) Moderate Complex Actor: Supports 2 or 3 transactions and assigned value is 4.
5) More Complex Actor: Finally, An Actor that interacts the system using Web Pages or Graphical User Interface (GUI) is called as More Complex Actor. The Corresponding weight is 5.

Unadjusted Use Case Weight (UUCW) : In EUCP, like Actors the Use Cases are also categorized into Simple, Advance Simple, Average, Complex and Advance Complex Use case. Assigning the Weights for the Use Cases depends on their transaction. If the number of transaction are at most three (<=3) and uses only one date base object then it is Simple Use Case. In Advance Simple Use case the number of transactions are more than or equal to 3 and use of stored procedures and corresponding weight is 7. If the transactions are Between 4 to 7 and assigned weight is 10 then it is called as Average. Complex Use Cases consist of more than 7 transactions and weight is also 15. If the transactions are greater than 7 and avail stored procedures then it is called as Advance Complex Use case and assigned weight is 17

Table1: Unadjusted Use Case Weights

Type of Use Case	No. of Transactions	Stored Procedures used	Corresponding Weight
Simple	At Most 3	No	5
Advance Simple	>=3	Yes	7
Average	4 – 7	No	10
Complex	>7	No	15
Advance Complex	>7	Yes	17

A) Calculation of Weights and Points:

UAW=∑ Simple Actor * 1 + ∑ Average Actor * 2 + ∑Average Complex Actor * 3+ ∑ Moderate Complex Actor *4 + ∑ More Complex Actor * 5 (1)

UUAW=∑ Simple Use Case* 5 + ∑ Advance Simple Use Case * 7 + ∑ Average Use Case * 10+ ∑ Complex Use Case * 15 + ∑ Advance Complex Use Case * 17 (2)

$$TCF = 0.6 + (0.01 \times \sum_{i=1}^{13} TF_i \times W_i)$$

$$ECF = 1.4 + (-0.03 \times \sum_{i=1}^{8} EF_i \times W_i)$$

Unadjusted Use Case Points (UUCP) is a size of the software and it is calculated as a sum of UAW and UUAW

UUCP= UAW+UUAW (3)

EUCP=UUCP*TCF*ECF (4)

B) Calculation of Environmental Factor and Technical Complexity factor

For the development of software project, functional as well as non-functional requirements are also essential. Functional requirements are represented using the Use case diagrams and Non-Functional requirements can be assed using the 13 technical factors and 8 Environmental parameters. Every parameter gets weights, from 0 to 5 where "0" suggests that the developer has no experience on the comparing parameter. On the off chance that the relegated weight of the parameter is "5" then the developer is considered as master in the particular domain of developing the application.

Table2: Technical factors

TCF$_i$	Description of Factors Contributing to Complexity	Weights
TF1	Distributed System	2
TF2	Application Performance Objectives	1
TF3	Client or End Users Efficiency	1
TF4	Complex Internal Processing	1
TF5	Reusability	1
TF6	Easy Installation	0.5
TF7	Portability	2
TF8	Changeability	1
TF9	Concurrency	1
TF10	Special Security Features	1
TF11	Provide Direct Access for Third Parties	1
TF12	Special User Training Facilities	1
TF13	Usability	0.5

Table 3: Environmental Factors

ECF$_i$	Factors Contributing to Efficiency	W$_i$
EF1	Familiar with objector	2
EF2	Part-Time workers	-1
EF3	Analyst Capability	0.5
EF4	Application Experience	0.5
EF5	Object Oriented Experience	1
EF6	Motivation	1
EF7	Complex Programming Language	-1
EF8	Stable Requirements	1

III. PERFORMANCE MEASURES

The execution of the diverse models can be estimated by availing the associated assessment measures. In this work, six different assessment criteria have been used, which are MMRE [15], MMER[15], RMSE[15], VAF[17] and PRED(x)[15]

- MRE (Magnitude of Relative Error to estimate): It is the refinement amongst actual and estimated effort with contrast with actual effort [15].

$$MRE = \frac{\text{ActEffort} - \text{EstEffort}}{\text{ActEffort}}$$

- MER (Magnitude of Error Relative): It is the refinement amongst actual and estimated effort with contrast with estimated effort

$$MER = \frac{\text{ActEffort} - \text{EstEffort}}{\text{EstEffort}}$$

MMRE (Mean of Magnitude of relative Error): It calculates the average of MRE for observed project data (N)

$$MMRE = \frac{1}{N} \sum_{1}^{N} MRE$$

- MMER (Mean of Magnitude of error relative): It calculates the average of MER for observed project data (N).

$$MMER = \frac{1}{N} \sum_{1}^{N} MER$$

- PRED (**n**): Forecast at level **n** is characterized as the % of projects that have absolute relative error not as much as **n**. For example, PRED (25) = 50 specifies that 50% of the projects fall within 25% error range.

- Variance Accounted For (VAF):

$$VAF(\%) = \left(1 - \frac{Var(Acteffort - Esteffort)}{Var\ Acteffort}\right) \times 100$$

- RMSE (Root Mean Squared Error):

$$RMSE = sqrt(mean(Acteffort - Esteffort)^2))$$

III. PROJECT DATASET

The size metrics used in the Datasets like COCOMO, NASA etc are either SLOC (KLOC) or Functional Points and these are not appropriate for the anticipated work [19]. This examination work is a push to propose a refined model based Enhanced Use case Point (EUCP) for predicting the software effort .The dataset has taken the part of the Use-Case based Effort Estimation Data Base (UCEEDB) [16] and have taken five student projects data into consideration [15]. UCEEDB contains the 14 projects data those are constructed by software development companies or students of Poznam University of Technology in Poland [16]. All the data have been transformed to EUCP with the calculation of UUCP, UUCW, TCF and ECF.

IV. RESEARCH METHODOLOGY

The software effort estimated as function of the size of project in the UCP method. Size of the project is measured using the Use Case Points and effort is expressed in man-hours. There are several effort prediction models are available in UCP method while selecting the functions. In this paper, EUCP method is considered to predict the effort of the software development. EUCP is extended the scope of previous UCP model with tailoring the categories and assigned weights of the Actors and Use Cases.

A) Karner's Model

It is very basic model to connect between size and effort. It is a linear model and he predicted the time needed for constructing 1 UCP is in between 20 to 28 man-hours [1]. Karner's model suffers from many faults and is not accurate. Karner's method is not used by many companies for their early stage effort prediction.

B) (ANFIS) Adaptive Neuro Fuzzy Inference System :

The Soft Computing Techniques or Machine Learning Techniques is a creative region which is delivering reliably exact results contrasted with the traditional methodologies. Soft Computing procedures are appropriate in circumstances where there exists imprecision, instability, unclearness and nonattendance of clarity. Soft computing is a consortium of methodologies with Fuzzy Logic, Artificial Neural Networks, Neuro Fuzzy Systems, Genetic Algorithms, Evolutionary Computation [17]. The main concentration is on Adaptive Neuro Fuzzy Inference Systems (ANFIS). ANFIS is a multilayer feed forward network which integrated the principles of neural network and fuzzy logic learning algorithms into a single framework [20]. In this paper, the function genfis2 of MATLAB is used for better prediction of the software development effort. The function genfis2 produces a Sugeno-type FIS structure utilizing subtractive clustering and requires separate arrangements of input and output data as input arguments. At the point when there is only one output, genfis2 might be utilized to produce an initial FIS for anfis training. When there is only one output, genfis2 may be used to generate an intial FIS for anfis training. Genfis2 accomplishes this by extracting a set of rules that models the data behavior.

TABLE 4. COMPARISON RESULTS

S.No	Evaluation Criteria	KARNER Method	Proposed Method
1	MMRE	65.2916	27.0428
2	RMSE	1075083	188362
3	MMER	44.81	12.51
4	VAF	28.61	82.63
5	PRED(25)	36.8421	84.2105
6	PRED(50)	63.1579	89.4737
7	PRED(75)	84.2105	100
8	PRED(100)	89.4737	100

V. CONCLUSION

In the literature, relation between KLOC/SLOC, FPA and UCP for effort estimation have been investigated. In the proposed work, evaluation of the ANFIS and Karner's method based on EUCP for two different datasets like UCEEDB[16] and Five sample student projects[15] analyzed and compared. Assessment criteria's like MMER, MMRE, RMSE, VAF and PRED (n) shown for the two techniques. ANFIS provides best results for all performances measure compared to Karner's. Forecasting the accurate and effective Effort in the early stage of software development will help the software project managers in planning and organizing the software development activities successfully.

REFERENCES

[1] A. B. Nassif, L. F. Capretz and D. Ho, "Enhancing Use Case Points Estimation Method using Soft Computing Techniques," Journal of Global Research in Computer Science, vol. 1, pp. 12-21, 2010.

[2] A. B. Nassif, L. F. Capretz and D. Ho, "Estimating software effort based on use case point model using sugeno fuzzy inference system," in 23rd IEEE International Conference on Tools with Artificial Intelligence, Florida, USA, 2011, pp. 393-398.

[3] D. Eck, B. Brundick, T. Fettig, J. Dechoretz and J. Ugljesa, "Parametric estimating handbook," The International Society of Parametric Analysis (ISPA), Fourth Edition. 2009.

[4] J. Lynch. (2009, Oct.). Chaos manifesto. The Standish Group. Boston. [Online]. Available: http://www.standishgroup.com/newsroom/chaos_manifesto.php.

[5] A. Albrecht, "Measuring application development productivity," in *IBM Application Development Symp.* 1979, pp. 83-92.

[6] C. R. Symons, "Function Point Analysis: Difficulties and Improvements," *IEEE Trans. Software Eng.*, vol. 14, pp. 2-11, 1988

[7] G. Karner, "Resource Estimation for Objectory Projects," *Objective Systems*, 1993.

[8] E. Mendes, N. Mosley and I. Watson, "A comparison of case-based reasoning approaches," in *Proceedings of the 11th International Conference on World Wide Web*, Honolulu, Hawaii, USA, 2002, pp. 272-280.

[9] M. Jørgensen, "Forecasting of software development work effort: Evidence on expert judgement and formal models," *International Journal of Forecasting*, vol. 23, pp. 449-462, 2007.

[10] L. C. Briand and I. Wieczorek, "Resource Estimation in Software Engineering," *Encyclopedia of Software Engineering*, vol. 2, pp. 1160-1196, 2002.
[11] B. W. Boehm, *Software Engineering Economics.* Prentice-Hall, 1981.
[12] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering*, vol. SE-4, pp. 345-361, 1978.
[13] D. D. Galorath and M. W. Evans, *Software Sizing, Estimation, and Risk Management.* Boston, MA, USA: Auerbach Publications, 2006
[14] G. Karner, "Resource Estimation for Objectory Projects," *Objective Systems*, 1993
[15] Meenakshi Saroha, Shashank Sahu "Software Effort Estimation Using Enhanced Use Case Point Model'' International Conference on Computing, Communication and Automation (ICCCA2015)
[16] Jovan, Sofizia, Marko and dragan " Enhancing Use Case Point estimation method using fuzzy algorithms" 23rd Telecommunications forum TELFOR 2015, Serbia, Belgrade, November 2r4-26, 2015
[17] Rama Sree P, Prasad Rededy P.V.G.D, Sudha K.R " Hybrid Neuro-Fuzzy System for Software Development Effort estimation" Rama Sree P et al. / International Journal on Computer Science and Engineering (IJCSE), Vol. 4 No. 12 Dec 2012, 1924-1932
[18] Shashank Mouli Satapathy , Barada Prasanna Acharya, Santanu Kumar Rath"Early stage software effort estimation using random forest technique based on use case points" IET Software, Research Article
[19] Jye-Shing Roger Jang "Adaptive – Network- Based Fuzzy Inference System" IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. 23, NO. 3, MAYIJUNE 1993
[20] Akshit Peer Ruchika Malhotra "Application of Adaptive Neuro-Fuzzy Inference System for Predicting Software Change Proneness" 978-1-4673-6217-7/13/$31.00_c 2013 IEEE, 2026
[21] B. Nassif, L. F. Capretz and D. Ho "Software Effort Estimation in the Early Stages of the Software Life Cycle Using a Cascade Correlation Neural Network Model" 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing.
[22] Mohammad Azzeh, Ali Bou Nassif "Fuzzy Model Tree For Early Effort Estimation" 12th International Conference on Machine Learning and Applications, 2013,117-121
[23] Roger S. Pressman "Software Engineering, A practitioner's approach