International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 17 (2015) pp 37353-37358


# An Accurate Effort Estimation using Least Effort Multipliers based on Fuzzy Estimation Algorithm for Software Development

**E.KARUNAKARAN,**
*Department of Computer Science and Engineering*
*Pondicherry Engineering College, Puducherry,*
*INDIA,* ***ekaruna@pec.edu,***

**N.SREENATH**
*Department of Computer Science and Engineering*
*Pondicherry Engineering College, Puducherry,*
*INDIA, nsreenath@pec.edu*

**Abstract**
Software companies are very keen in delivering accurate and reliable software products to the end users especially by minimizing software development effort. Software project managers also have designed various estimation techniques for improving the efficiency of the product by reducing the cost. Hence, there arises a need for developing a hybrid mechanism that enables accurate estimation of effort. This paper presents an accurate Least Effort Multipliers based on Fuzzy Estimation Algorithm (LEMFEA) that empirically studies the impact of five effort multiplier factors like project type (T), programmers skill (S), software language used (L), database used (D) and criticality(C) of the software based on fuzzy-based function point analysis. This fuzzy-based estimation algorithm utilizes a fuzzy-based estimate to identify the uncertainty of the software size with the aid of triangular fuzzy set and then a function point analysis is incorporated by the five effort multiplier factors with the effort estimation for accuracy. Furthermore, experimentation is carried with different project data sets and the result infers that the proposed LEMFEA algorithm not only improves the accuracy of effort estimation but also increases the reliability of the software product. The proposed LEMFEA estimation model is superior in estimating the efforts than any other functional point analysis model available in the literature. In addition, LEMFEA algorithm tolerates imprecision, offers transparency in the prediction process and possess the capability of adapting to changing environment depending on the dynamic availability of new data.

**Keywords:** Estimation, Effort Estimation, Sizing the software, COSMIC full function, fuzzy-based functional point, Software Measurement, eXtreme software size Unit (XU).

## 1. Introduction

Software effort estimation is one of the significant steps in software project management process since the success or failure of the software project highly depends upon the accuracy of the effort and schedule estimates [1]. Effective software project estimation process is one of the most challenging activities in the software development since proper project planning, monitoring and controlling cannot be done efficiently without accurate and reliable estimates [2]. The criticalities associated with software effort estimation are, i) effort estimation process must be done in earlier phase of software planning and development, ii) although number of methods and metrics are available for effort estimation, when the size of the software grows exponentially, all those existing methods fail to produce accurate effort estimation and iii) software effort estimation process lacks with reliable and secure methods which are invulnerable to attacks and failures [3].

## 2. Related Work

Researchers have proposed number of techniques for accurate estimation of cost and effort involved in software project by considering the system security [4]. In many of the previous research works, authors have proved that the improvement of the accuracy of the highly dependent on software estimation and also the fuzzy-based function point analysis is incorporated to overcome the unpredictability and riskiness in effort estimation [5]. Some of the software estimation approaches are detailed below.

In [6], authors presented a fuzzy-based function point analysis method for handling ambiguous and linguistic inputs for estimating the effort and cost involved in software project. In contrast, the homogeneous data sets were considered for software effort estimation in [7] which results in accurate and reliable software estimates. In this work, the comparative analysis is done by considering both homogeneous and irrelevant or disordered data sets to prove the significance of the ordered set of input data for effort estimates. In reference [8] authors have proposed a novel fuzzy-based framework for managing imprecision and uncertainty problem in effort estimation process.

Similarly, in [9] authors have investigated a hybrid methodology by integrating neuro-fuzzy and SEER-SEM techniques that can effectively functions with various other algorithmic models for effort estimation. Further, the work presented in reference [10], is Enhancing Software Sizing Adjustment Factors which effectively estimates the software efforts by predicting size of the software. Moreover, in [11], authors contribute an enhanced analogy-based approach based on extensive dimension weighting and this method shows experimentally evaluated results for project efforts. Recently,

a work suggested by the researchers in [12] proves that the change in standard function points analysis may improve the accuracy in effort estimation by reducing the ambiguity. Recently, in work [13], authors demonstrated that effort estimation can be performed by incorporating soft computing technique to handle uncertainty in input dataset. The main limitation associated with all the aforementioned existing works is that, they all focus of eliminating uncertainty and complexity involved in input data with least concentration with development process involved in effort estimates. Many of the existing research works concentrates in accuracy issue than the other performance related issues like security.

## 2. Extract of the Literature

From the literature review conducted, the following shortcomings are identified as below:

a) Generally, the similarities between these studies focus on the data sets or the initial phase of the estimation but do not concentrate on the development phase of the effort.

b) Most of the methods use the fuzzy logic to handle imprecision in the data sets but does not focus on performance factors.

These limitations induced us in proposing a Least Effort Multiplier based Fuzzy Estimation Algorithm for effort estimation.

## 3. Proposed Work

### 3.1 LEMFEA Algorithm and Model

The proposed Least Effort Multiplier based Fuzzy Estimation Algorithm (LEMFEA) is an enhanced version of FFPA-PSR (fuzzy-based function point analysis with performance metrics, security, and reliability factors) algorithm that has been proposed for improving the accuracy of the software effort estimation. The proposed LEMFEA algorithm uses fuzzy logic to frame rules based on the classification of the attributes like project type (T), programmers skill (S), software language used (L), database used (D) and criticality(C) required for the estimation. The performance factors are mainly integrated for enabling the estimation model to estimate effort in an accurate and automated way. Since, accurate effort is considered to be more significant because inaccurate and insecure estimation of software effort estimation leads to drastic deviation in the expected and actual budget.

#### 3.1.1 Least Effort Multiplier based Fuzzy Estimation Algorithm (LEMFEA)

Step 1: Elucidate the requirement specification of the project to be estimated.
Step 2: Initialize the function points for analysis.
Step 3: Classify the identified function point using least multiplier factors and integrate them.
Step 4: Use triangular membership function for estimating the impact of utilised five least multiplier factors.
Step 5: Analyse the fuzzy rule generated using cross over mechanism.
Step 6: Modify and enhance the Value Adjustment Factor (VAF) with respect to least multiplier factors.
Step 7: Manipulate the fuzzy function point through Unadjusted Function Point (UFP) and Value Adjustment Factor (VAF).
Step 8: Determine the performance metrics by estimating the precision value.
Step 9: Modify the precision value based on the estimation.
Step 10: Calculate the enhanced effort estimation based on the fuzzy-based function point analysis.
Step 11: Examine the accuracy of LEMFEA through real time data.
Step 12: The model is implemented if the results are superior with these fuzzy rules. Else, Create new fuzzy rules.

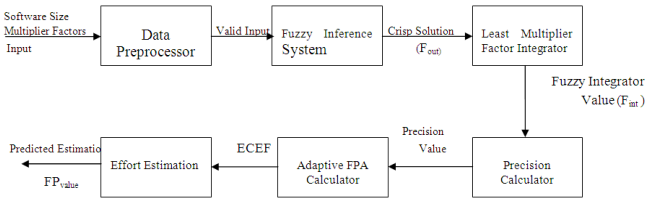


**Fig. 1 Block diagram of the proposed LEMFEA system**

The input of the developed model is the software size with the five least multiplier factors and the output is the estimated effort. This model incorporates four different entities for effort estimation viz., i) Fuzzy Inference System, ii) Least Multiplier Factor Integrator, iii) Precision Calculator, iv) Adaptive FPA Calculator and v) Effort Estimator. Further, Fig. 1 depicts the block diagram for the proposed LEMFEA algorithm.

**i) Fuzzy Inference System**

Software development process is a complex process that necessitates the interaction of factors of function point analysis like External inputs, External interfaces, External inquiries, External outputs and Internal logical files. Further, based on the experience in the software development process, it is inferred that least amount of multipliers are enough for effort estimation. The significant factors that are identified to influence the software development process significantly are Type of the software, Programmers skill, Language or tool used, Database used and Criticality of the software respectively designated as T, P, L, D and C. Furthermore, these five factors are represented by the five tuples <T, P, L, D, C>. In addition, the other factors are either included in any one of these five factors or it has negligible impact in the software development process. Finally, the ratings of these factors are designated as 'TOLERABLE', 'SIGNIFICANT' and 'SENSITIVE'. In order to handle the dependencies and precision of the factors, they are fuzzified to improve the accuracy [ ]. But, this fuzzification is difficult to achieve. Thus, fuzzy if –then rules are generated to tackle this situation. The fuzzy rules used are:

IF complexity of development is MINIMUM and the assigned weight is also MINIMUM, THEN the fuzzy function point is TOLERABLE;

IF complexity of development is MODERATE and the assigned weight is also MODERATE, THEN the fuzzy function point is SIGNIFICANT;
IF complexity of development is MAXIMUM and the assigned weight is also MAXIMUM, THEN the fuzzy function point is SENSITIVE;
The outputs of each fuzzy rule are normalized by defuzzification that converts fuzzy output into crisp solutions ($F_{out}$) for required output by using the following equation (1).

$$F_{out} = \frac{\sum(F_{in} * V_{in})}{F_{in}} \quad (1)$$

Where,
$F_{in}$ - Mean expected value of input factor.
$V_{in}$ - Impact factor of each input factor on a six-point scale (0-5)

**ii) Least Multiplier Factor Integration**
In the next step, the outputs of the crisp solutions are integrated with the least multiplier factor. The multipliers values that are considered in this LEMFEA algorithm is directly taken from the research work [14], which has been derived from twenty different real time project data. Further, the impact of each multiplier is analyzed using table 1 that gives the multiplier values for different software types and the programmer's skill to decide the effort required to develop the software. Furthermore, table 2 gives the multiplier values for different software types with respect to a software language or tool used for developing the software. It also shows the effort estimation multiplier values between some of the software used (MIS, WEB, TELECOMMUNICATION) and the languages like VB, JAVA, C, PHP, VC++ and.NET. Similarly, table 3 and 4 presents the multiplier value that provides the complexity function weights between database with software type and also analyses criticality with software types respectively.

**Table 1: Effort multiplier values (Software Type Vs Programmer's Skill)**

| Developer \ Software Used | Beginner | Skilled | Expert |
|---|---|---|---|
| MIS | 2.0 | 1.1 | 0.6 |
| WEB | 3.2 | 1.5 | 1.0 |
| TELE | 3.5 | 2.1 | 1.3 |

**Table 2: Effort multiplier values – (Software Type** Vs **Software or tool Used)**

| Software Used | VB | JAVA | C | PHP | VC++ | .NET |
|---|---|---|---|---|---|---|
| MIS | 1.0 | 1.3 | 1.5 | 1.4 | 1.7 | 1.2 |
| Web | 1.2 | 1.5 | 2.1 | 1.4 | 2.2 | 1.7 |
| Tele Comm. | 2.2 | 1.9 | 2.0 | 1.7 | 2.0 | 1.9 |

**Table 3: Effort multiplier values – (Software Type** Vs **Database Used)**

| Database Used | Oracle | Access | MySql |
|---|---|---|---|
| MIS | 1.0 | 0.75 | 0.9 |
| Tele Comm. | 1.3 | 1.4 | 1.2 |
| Web | 1.25 | 1.35 | 1.2 |

**Table 4: Effort multiplier values – (Software Type** Vs **Criticality of the project)**

| Criticality | Catastrophic | Moderate | Least Significant |
|---|---|---|---|
| MIS | 1.0 | 0.75 | 0.9 |
| Tele Comm. | 1.3 | 0.7 | 0.3 |
| Web | 1.2 | 0.85 | 0.6 |

The least multiplier factors are integrated together with the fuzzy crisp value to obtain Fuzzy integrator value ($F_{int}$ ) by using the equation (2).

$$F_{int} = F_{out} * A_{ls} \quad (2)$$

where, $F_{out}$ and $A_{ls}$ represents fuzzy output and least multiplier output respectively.

**iii) Precision Calculation**
The performance of any software system depends on the 5 important factors like speed, accuracy and latency. Hence, the precision value is estimated through equation (3).

$$FP_{value} = 0.01 * \sum_{f=1}^{5}(P_f * C_f). \quad (3)$$

where,
$P_f$ - Performance factor
$C_f$ - factor of complexity

**iv) Adaptive FPA Calculator**
The effort required for estimating effort (EEF) depends on the multiplication of two factors called Unadjusted function points (UFP) and value adjustment factor (VAF) as depicted in equation (4).

$$EEF = UPF * VAF \quad (4)$$

Once the precision and functional point count are estimated, then the enhanced or change in effort estimation (ECEF) is calculated using equation (5).

$$ECEF = EEF + FP_{Value} \quad (5)$$

**v) Effort estimation**
Finally, based on the value of ECEF, the effort estimation of the software project is assured.
This estimated effort depends on a five point scale of influence that depends on the role of each factor considered for effort estimation as given below:
0-Very Low
1-Low
2-Normal

3-High
4-Very high

## 4. Results and Discussions

The proposed LEMFEA estimation technique is developed in Java script under windows environment and validated with real project data sets. The effort estimation data of twenty implemented software projects of 2014 is used for testing. At the same time, the proposed LEMFEA estimation technique is compared with COCOMO-II, Calibrated COCOMO-II [15-18]. The comparative results for estimated efforts are presented in Fig.2.

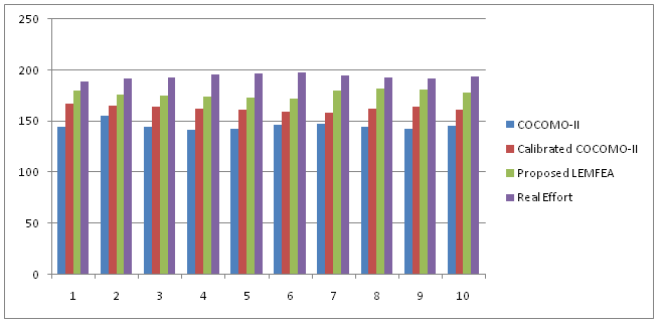


**Fig. 2: Effort estimation chart**

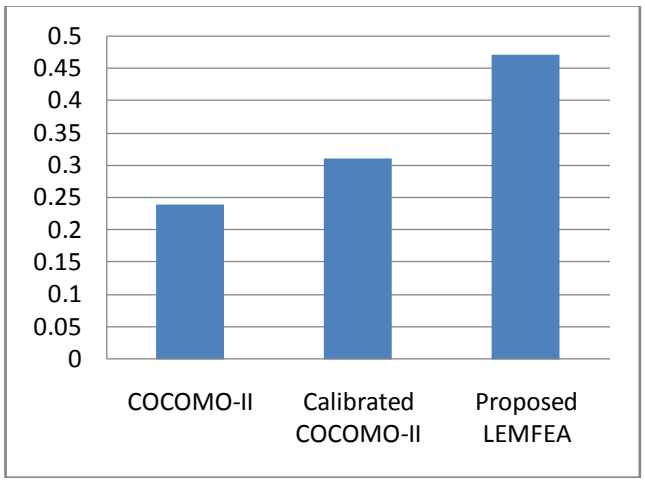


**Fig.3: Effort estimation chart based on PRED**

From Fig.3, it is transparent that the proposed LEMFEA algorithm estimates effort values that are very closer to the real value estimations. It also infers that the accuracy of the LEMFEA algorithm is maximum, but practically it is proved based on the popularly known performance evaluation metrics like Mean Magnitude Relative Error (MMRE) and Prediction rate (PRED). The Mean Magnitude Relative Error (MMRE) and Prediction rate (PRED) are calculated based on the ratio of actual number of observations to the number of estimates[19-21] generated by the model as presented through Fig.3 and Fig.4 respectively.

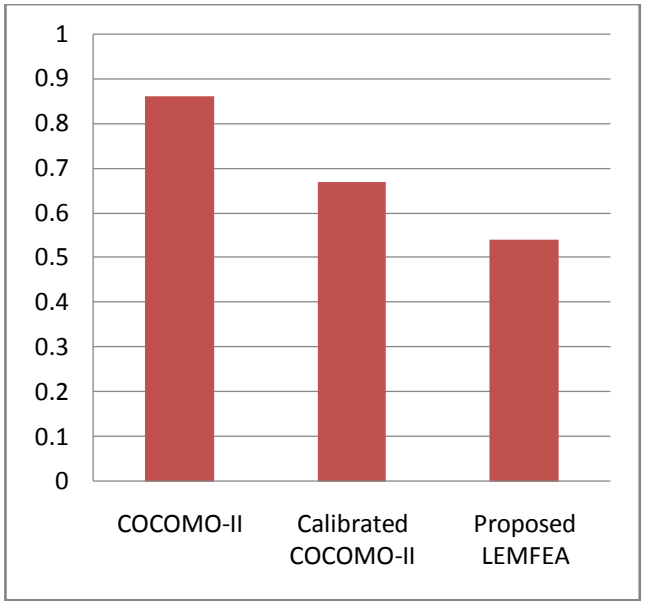


**Fig. 4: Effort estimation chart based on MMRE**

Further, the Mean Magnitude Relative Error (MMRE) is calculated based on the Mean Relative Error factor estimated by the ratio of deviation between the Real Effort and Calculated Effort to the Real Effort estimated. Furthermore, the Prediction Accuracy (PRED) is also manipulated through equation (6) as,

$$PRED = \lambda / N \text{.....} \quad (6)$$

Where, '$\lambda$' denotes the number of projects estimated and '$N$' indicates the maximum number of instances of all estimates.
From Fig.3, it is evident that the PRED value of the proposed LEMFEA technique is comparatively maximum when compared to the benchmark estimation techniques like COCOMO-II, Calibrated COMO-II. From Fig.4, it is evident that the MMRE value of the proposed LEMFEA technique is comparatively minimum when compared to the benchmark estimation techniques like COCOMO-II and Calibrated COCOMO-II.
Likewise, the proposed estimation technique is compared with the existing models in terms of accuracy that predicts a lower MMRE value and higher prediction value. This analysis based on accuracy is depicted through Fig.5.

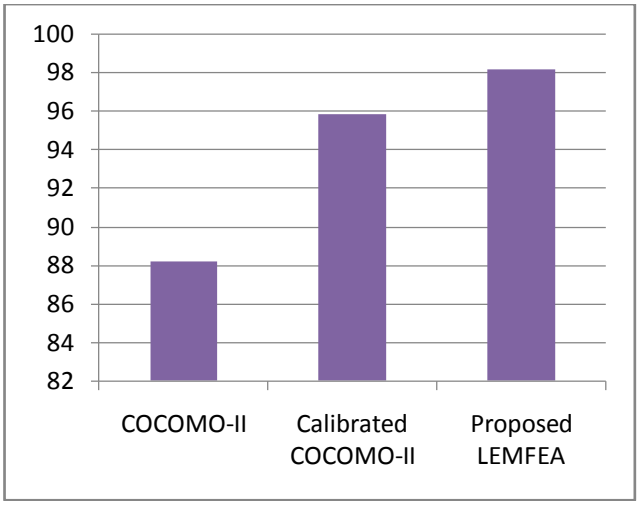


**Fig.5 : Effort estimation chart based on Accuracy**

Fig.5, proves that accuracy value of estimation (expressed in percentage) of the proposed LEMFEA technique is highly superior to the compared benchmark estimation techniques like COCOMO-II, and Calibrated COCOMO-II. Furthermore, the estimation models are incorporated for effort estimation either through training or through testing. Fig.6 presents the comparative analysis of estimation through training and testing respectively. It also portrays that, the proposed LEMFEA technique is comparatively minimum in terms of MSE, MAE and RMSE but possesses high VAF than the compared benchmark estimation techniques like COCOMO-II and Calibrated COCOMO-II. In addition, the proposed estimation model is also validated through chi-square test and the evaluation reveals that COCOMO II model and the calibrated COCOMO II model as worst fit as presented in table 5.

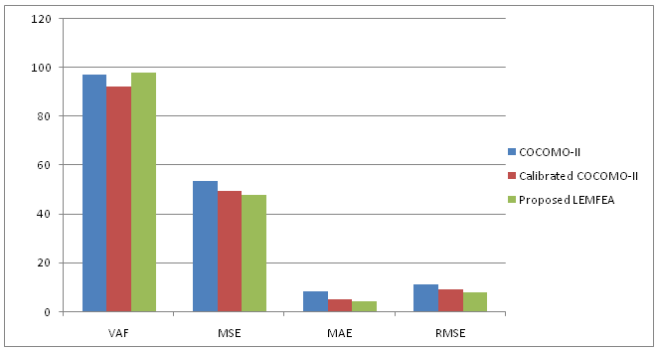


**Fig. 6 : Effort estimation chart based on VAF, MSE, MAE and RMSE based on training**

**Table 5 Chi-Square based fitness validation for the estimation models**

| | Model I (COCOMO II) | Model II (Calibrated COCOMO II) | Model III Proposed (LEMFEA) |
|---|---|---|---|
| $\chi^2$ | 1.12 | 1.34 | 1.87 |
| $\mu$ | 1.39 | 1.67 | 1.96 |
| $\sigma$ | 1.1 | 1.3 | 1.6 |

**Table 6 Performance Comparison based on percentage of improved accuracy**

| Model | Accuracy | Improved accuracy (in %) |
|---|---|---|
| COCOMO II | 85.12 | 6.72 |
| Calibrated COCOMO II | 95.67 | 9.34 |
| Proposed (LEMFEA) | 98.65 | 12.33 |

Finally, Table 6 shows the performance comparison based on percentage of improved accuracy. In addition, the choice of factors like project type (T), programmer's skill (S), software language (L), database used (D) and criticality(C) of the software has greater influence on the accuracy of the proposed LEMFEA estimation model. It also improves the accuracy of estimation by an average of 9.34 % than the function point model like COCOMO-II and Calibrated COCOMO-II, while it improves the accuracy by 12.33% than the existing functional point's model.

## 5. Conclusion

This paper has presented automated hybrid tool called LEMFEA estimates the effort involved in software project in an accurate and reliable way. The proposed hybrid model is evaluated by means of integrated datasets with ten different project domains. The proposed hybrid model shows better results than the existing models in terms of accuracy in effort estimation, reliable and secures estimation factors which may be utilized in highly critical applications. This model is highly efficient to handle uncertainty in input data set for effort estimation which is obtained due to incomplete requirement information for software project. The proposed model efficiently estimates the effort within the time and budget frame which is one of the much needed requirements of the software organizations. Our future plan of research is to extent this work for assessing risks involved in software project development.

## References

[1] J. F. Vijay and C. Manoharan, "Initial hybrid method for analyzing software estimation, benchmarking and risk assessment using design of software," Journal of Computer Science, vol. 5, no. 10, pp. 717–724, 2009.

[2] X. Huang, D. Ho, J. Ren, and L. F. Capretz, "A soft computing framework for software effort estimation," Soft Computing, vol. 10, no. 2, pp. 170–177, 2005.

[3] J. Sedlackova, "Security factors in effort estimation of software projects," Information Sciences and Technologies Bulletin of the ACM Slovakia, vol. 3, no. 2, pp. 12–17, 2011.

[4] V. V. Manoj and R. S. Kumar, "A novel interval type-2 fuzzy software effort estimation using Takogi-sugeno fuzzy controller," International Journal of Modern Engineering Research, vol. 2, no. 5, pp. 3245–3247, 2012.

[5] M. Azzeh, D. Neagu, and P. I. Cowling, "Analogy-based software effort estimation using Fuzzy numbers," Journal of Systems and Software, vol. 84, no. 2, pp. 270–284, 2011.

[6] I. Attarzadeh and S. Hockow, "Improving the accuracy of software cost estimation model based on a new fuzzy logic model," World Applied Sciences Journal, pp. 177–184, 2010.

[7] H. K. Verma and V. Sharma, "Handling imprecision in inputs using fuzzy logic to predict effort in software development," in Proceedings of the IEEE 2nd International Advance Computing Conference (IACC '10), pp. 436–442, Patiala, India, February 2010.

[8] M. A. Ahmed and Z. Muzaffar, "Handling imprecision and uncertainty in software development effort prediction: a type- 2 fuzzy logic based framework," Information and Software Technology, vol. 51, no. 3, pp. 640–654, 2009.

[9] W. L. Du, D. Ho, and L. F. Capretz, "Improving software effort estimation using neuro-fuzzy model with SEER-SEM," Global Journal of Computer Science and Technology, vol. 10, no. 12, pp. 52–64, 2010.

[10] H. Moussa, G. H. Galal-Edeen, and A. Kamel, "Enhancing software sizing adjustment factors," in Proceedings of the 4th International Conference on Intelligent Computing and Information Systems (ICICIS '09), pp. 438–444, ACM, Cairo, Egypt, 2009.

[11] M. Aver and S. Biffi, "Increasing the accuracy and reliability of analogy-based cost estimation with extensive project feature dimension weighting," in Proceedings of the International Symposium on Empirical Software Engineering, pp. 2165–2170, 2004.

[12] M. A. Al-Hajri, A. A. A. Ghani, M. N. Sulaiman, and M. H. Selamat, "Modification of standard function point complexity weights system," Journal of Systems and Software, vol. 74, no. 2, pp. 195–206, 2005.

[13] M. Senthil Kumar and B. Chidambara Rajan, "Impact of performance metrics in software effort estimation using function point analysis," Information, pp. 2253–2266, 2014.

[14] E.Karunakaran and N.Sreenath, "A Method to Effort Estimation for XP Projects in Clients Perspective", International Journal of Applied Engineering Research, ISSN-0973-4562, Vol.10,No.7, 2015, pp.18529-18550.

[15] Banker, R., H. Change, and C. Kemerer, "Evidence on Economies of Scale in Software Development," Information and Software Technology, vol. 36, no. 5, 1994, pp.275-828.

[16] Shepperd, M. & Schofield, M, "Estimating Software Project Effort Using Analogies, IEEE Transactions on Software Engineering, vol. 23, no. 12, 1994, pp. 736-743.

[17] Selby, R., "Empirically Analyzing Software Reuse in a Production Environment," in Software Reuse: Emerging Technology, W. Tracz (Ed.), IEEE Computer Society Press, 1988, pp.176-189.

[18] Xiao Xiao and Tadashi Dohi. "Wavelet Shrinkage Estimation for Non-Homogeneous Poisson Process Based Software Reliability Models," IEEE Transaction on Reliability vol.62, no.1, March 2013.

[19] S.Basu and N.Ebrahimi. "Bayesian software reliability models based on martingale processes. Technometrics," 2003, vol.45, no.2 pp.150-158.

[20] AGandy, U.Jensen. "A non-parametric approach to software reliability," Appl. Stochasti Models Business Ind., 2004 vol.20, no.1, pp3-15.

[21] S.P. Wilson and F.J.Samaniego. "Nonparametric analysis of the order-statistic model in software reliability," IEEE Transactions on Software Engineering, 2007, vol.33, no.3, pp 198-208.