Software Project Effort Duration and Cost Estimation using Regression Testing and Adaptive Firefly Algorithm (AFA)

B.M.G.Prasad, P.V.S. Sreenivas, C.Veena

Abstract: ***Software project estimation is the process of defining the quality of the software projects in terms of effort, duration and cost. The better software construction should take less effort with ensured short duration and cost. Estimating the software project EDC estimation is a more difficult task which is focused by various researchers. COCOMO is found to be most successful software project cost estimation model. However this research work requires complete information for the decision making. And also this research method would lead to more computational overhead for the decision making process. This is focused and resolved in the proposed research methodology by introducing the new software EDC estimation process namely Regression Testing based Software EDC Estimation Technique (RTSEDCET). The experimental analysis is carried out on two datasets namely NASA 93 and COCOMO datasets. The proposed regressing testing model would generate the various test cases by comparing the attribute values of these datasets to predict the quality of the software in terms of effort, duration and cost. Based on these test case values, software project estimation can be done efficiently. In this work, adaptive firefly algorithm is utilized for the efficient test case generation which would combine the multiple attributes of the dataset to generation optimal test cases with the concern of ranking. The overall evaluation of the research work is conducted on the java simulation environment from which it is proved that the proposed research technique leads to ensure the optimal outcome than the existing research techniques.***

Keywords: ***Project quality, Effort, Duration, Cost, Regression testing, Multi attribute merging***

I. INTRODUCTION

Software development effort gauges are the reason for undertaking offering, planning and arranging [1]. These are basic practices in the product business, since poor planning and arranging frequently has sensational results [2]. The normal contention on the undertaking cost invades is vast. Boraso detailed that 60% of vast undertakings essentially invade their appraisals and 15% of the product ventures are never finished because of the gross misestimating of improvement exertion. Conveying a product item on time, inside spending plan, and to a concurred dimension of value is a basic worry for programming associations [3]. Precise assessments are critical for better arranging, checking and control [4]. Jones [5] proposes programming quality as "programming quality means being on time, inside spending plan, and addressing client needs". Then again, it is important to give the client or the designer association an early sign of the task costs [6]. As a result, extensive research consideration is presently coordinated at picking up a superior comprehension of the product advancement process and additionally building and assessing programming cost estimation instruments [7]. Thusly, there has been over the top focal point of research on estimation techniques from an assortment of fields. Statistical regression analysis is the most reasonable system to adjust execution models in a discovery mold [8]. Statistical regression analysis is a strategy which models the connection between a lot of info factors, and at least one yield factors, which are viewed as to some degree reliant on the contributions, based on a limited arrangement of information/yield perceptions. The evaluated model is basically an indicator, which, when nourished with a specific estimation of the info factors, restores a forecast of the estimation of the yield [9]. The objective is to get a solid speculation that implies that the indicator, aligned based on a limited arrangement of watched measures, can restore an exact forecast of the needy variable when a formerly concealed estimation of the free vector is introduced [10]. In different terms, this strategy expects to find and to evaluate, based on perceptions just, potential relationships between's arrangements of factors and utilize these connections to extrapolate to new situations [11]. The main contribution of this research method is to implement the software estimation framework which would measure the cost, effort and duration of the projects. This method utilizes the regression testing for the software effort estimation by generating the test cases from the datasets. Based on these test cases, software project EDC estimation is done efficiently. The proposed research method ensures the faster and accurate estimation values by adapting the regression testing based estimation. The performance of the regression testing is improvised by introducing the adapting firefly algorithm which will select the most optimal set of multiple attributes to form the test. The evaluation of this research work is carried out in the NASA 93 dataset and COCOMO dataset.

Manuscript published on 30 January 2019.
***** Correspondence Author (s)

B.M.G.Prasad, Research Scholar Department of Computer Science and Engineering PP.COMP.SCI&ENG.0064C Rayalaseema University Kurnool – 518007, Andhra Pradesh, India. (bmgprasad@gmail.com)
Dr.P.V.S. Sreenivas, Professor Department of Computer Science and Engineering, Sreenidhi Institute of Science, & Technology Hyderabad, Telangana, India – 501301. (pvassrinivas23@gmail.com)
C.Veena, Asst. Professor Department of Computer Science and Engineering Holy Mary Institute of Technology & Science Hyderabad, Telangana, India – 501301 (cveena30@gmail.com)

The overall organization of the research work is given as follows: In this section detailed introduction about the software effort estimation and the role of regression testing n software estimation is explained. In section 2 discussions about the various related research methodologies is explained whose main goal is to measure the software project estimation is done. In section 3, discussion about the proposed research methodology is given with the suitable examples and explanation. In section 4, performance evaluation of the proposed research methodology is given. Finally in section 5, overall conclusion of the proposed research methodology is given based on simulation outcome obtained.

II. RELATED WORKS

The arrangement of software cost estimation models and procedures are accounted for and classifications by Suri and Ranjan [12]. The most existing strategies for programming cost and exertion estimation will be estimation by Analogy, Expert Opinion, Putnam's Software Life-cycle Model (SLIM) and COCOMO. The similar examination of programming estimation strategies and the parts of every technique was talked about as relationship estimation needs authentic information which not accessible in all association, furthermore Delphi estimation strategy which is master assessment for venture determination is taken, thirdly SLIM Putnam programming exertion estimation mode at long last essential COCOMO which is great at snappy and early cost gauge for programming with impediment of precision because of absence of bookkeeping factors in task quality (cost driven) which middle COCOMO account the nitty gritty and impact of undertaking stages.

The conventional model [13] utilized Principal Component Analysis (PCA) with Artificial Neural Network (ANN) base of (COCOMO II) model to enhance the accuracy of cost estimation. The connection to the expense of assets that are basic amid programming advancement, So the blend of these models help to evaluate programming cost and exertion by compute the size and make a mixture apparatus. The PCA and ANN are the two models that are progressively exact; henceforth, they can be the answer for the cost estimation. In this manner, the consequence of PCA and ANN dependent on COCOMO are increasingly precise and increment the accuracy of estimation without declining the inconstancy. Algorithmic and Non-algorithmic strategies, Function point size, COCOMO, and ANN were consolidated by [14] to help in showing signs of improvement the rightness of programming estimate systems. The investigation consolidated the three methods as a mixture show and came about that the precision has been enhanced contrasted and COCOMOII and ANN. The advantages of the proposed model are the time turnaround and exactness. An Adaptive Learning Approach to Software Cost Estimation article by Reddy and Raju [15] proposes the utilization of back engendering neural systems as a way to deal with perform cost estimation for programming. A counterfeit neural system is relatively like the organic neural systems. Backpropagation neural system suits and enhances the COCOMO procedures. The model comprises of COCOMO dataset and the dataset for COCOMO NASA 2 that are utilized to prepare and test dependent on the recreation and models performed. Saroha and Sahu [16] assessed instruments and strategies for programming exertion estimation by utilizing use case point display. The survey talked about the estimation of the improvement procedures of programming by thought of exertion utilized in estimation as take more and expends a great deal of time. The survey demonstrates that the use of the UCP can be progressively successful and effective when contrasted with different models, for example, COCOMO and FPA. Improving COCOMO estimation by utilizing neural system proposed by [17]. The proposed arrangement looks to enhance productivity by improving the expectation exactness in the COCOMO show as a product exertion estimation framework applying the counterfeit neural systems. A multi-layer feed forward neural system used to continue the model in addition to its related parameters. The proposed system is splashed with a back engendering learning calculation preparing. Disappointment in mistakes estimation at the beginning times of programming items can be exorbitant with respect to costs and time examined by Gupta et al. [18]. The proposed guideline based experiment decrease procedure utilized choice table and it indicated blunders may emerge at later phases of the improvement of programming. Also, the mistakes in the underlying phase of SDLC with less exactness and effectively went to too are probably going to cost programming advancement organizations all in all a consider penny and time misfortune. Structures are considered as a spine of this investigation.

III. SOFTWARE PROJECT ESTIMATION BASED ON REGRESSION TESTING

The main contribution of this research method is to implement the software estimation framework which would measure the cost, effort and duration of the projects. This method utilizes the regression testing for the software effort estimation by generating the test cases from the datasets. Based on these test cases, software project EDC estimation is done efficiently. The proposed research method ensures the faster and accurate estimation values by adapting the regression testing based estimation. The performance of the regression testing is improvised by introducing the adapting firefly algorithm which will select the most optimal set of multiple attributes to form the test. The evaluation of this research work is carried out in the NASA 93 dataset and COCOMO dataset.

3.1. REGRESSION TESTING (RT)

This research work presents a novel means to denote multi-criteria techniques. In this research, a hybrid method is utilized for software project estimation by means of employing two criteria: 1) frequency based criteria and 2) a combinatorial criterion. The test cases are well-organized by one criteria and while the Average Percent of Faults Detected (APFD) does not improve one time a particular number of test cases are reached, changed to a second criteria.

This evaluation uses call center web application with seeded faults. In the presented hybrid criteria function is enhanced with the help of Adaptive Firefly Algorithm (AFA) for RT in the company of test cases function values.

3.1.1. Hybrid Test Criteria (HTC)

For performing RT tasks, this segment properly describes the three hybrid conditions. As hybrid criteria are obtained by incorporating individual criteria in diverse ways, the hybrid criteria are proposed using the functional pattern. Take that the test suite encompasses five test cases samples , those are in fact an ordered series of events; diverse measured features of these test cases, all-encompassing of the events(ε), statements(S), and branches(B) covered by them, their execution time, length(L), and faults (F) identified by them. It is presumed that these are denoted as matrices (such as, coverage matrix S for the statements covered) and in the vectors of values (i.e., L is represented the test cases length) without losing generality. The more information of test case examples is illustrated in existing work.

Representing Stand-Alone Criteria

It is begun by means of stating the existing traditional, non-hybrid criteria as functions. The vital components of formulation is mainly a function next () that considers three parameters, as the consideration is on test case prioritization,: 1) the test case order chosen thus far, 2) the entire test suite, 3) a matrix that encodes the association amongst the complete test cases in addition the metric that is being utilized to compute the ranked order for instance, statement coverage combined with a function g() for further coverage prioritization with the intention of being utilized for the computation. A common use of next () would begin with a blank series as first parameter; next () would recover the test cases with the maximum priority;

Functional formulation permits the fast recognition of alternative test case prioritization by means of emerging novel invocations as well as g() function that could “plug into” the entire prioritization structure. The g() function is constrained so as to function on an individual coverage matrix/vector. The entire coverage elements get covered by the test cases that are chosen before. Consequently, the latter suite, stored in y, as it exactly covers the similar coverage elements, becomes the reduced form of Suite. Instead, Test suite reduction needs fairly more involved modification. A non-greedy g() function is described with the intension that a reduced suite is obtained.

Representing Second-Order Criteria

The design is expanded to consider multiple criteria, and the developing term as second-order criteria. The conditions might be combined in quite a lot of means to produce a hybrid since it could be deduced. Currently, stooping back is essential as well as the viewpoint behindhand the creation of hybrids is to be taken for the test case prioritization. The basic idea behindhand and the hybrid is that it integrates diverse condition, providing, at each choice that test case must be selected consequent to the prioritization method, the powers of each fundamental individual criterion. A probable hybrid is described as the ordering of every individual criterion begin with the principal, and while the primary one gives ties in test case selection, utilizes the subordinate merely, to be exact, multitude of test cases meet the primary condition. This results in a ranking type of hybrid.

E.g., take that it is desired to order the tests by their capability in covering the summation of statement as well as branch coverage, to be precise that the most important test case covers numerous amount of statements as well as branches unified. This creates hybrid criteria that merge the criteria together in addition the mixture is employed all at once. As a final point, a third type of hybrid has to let a perfect choice amid multiple conditions dependent upon a selection function. In the latest work comprising merge and Rank formalizations. Rank: By perception, the conditions, which are being combined are ranked by the order of significance and after that used serially. Two sets of conditions are used in series. Rank based conditions, which are got for web applications, for instance base requests, name-value pairs, and series encompassing base requests with size two, and series encompassing base requests as well as name of size two are used. The principal invocation of the next () function is as follows

$$next([], T_1 \dots T_i, \dots T_n\},$$
$$Rank\left(\left(M, g_{mod-hgs}()\right), \left(base, g_{mod-hgs}()\right)\right)$$

here the literals M as well as base indicate the mapping amid the test cases as well as approaches, and base requests, respectively. The function $g_{mod-hgs}()$means their recognition of the revised Harrold, Gupta, Soffa (HGS) algorithm. Albeit the “HGS algorithm” was used with a single condition, their real work clearly describes the means wherein it is used with several amounts of conditions.

Similar kind of invocations is expressed while the method coverage is combined with the residual usage-based condition. Afterward, a subsequent call of next () would utilize the test cases that are noticed regarding method coverage as well as evaluate them to break the ties corresponding to base requests coverage. Lin et al approach is an application of Rank hybrid formalization, wherein, branch coverage was utilized as the primary criterion and then the secondary criterion is pair coverage. The primary invocation of the next () function is as follows

$$next([], T_1 \dots T_n\},\ \ Rank\left(\left(B, g_{m-hgs}()\right), \left(DU, g_{m-hgs}()\right)\right)$$

here B as well as DU state the mappings amongst the test cases and the branches, defuse pairs that are covered, respectively. For example ,to prioritize the test cases, one recognition of Rank could utilize the primary condition; the second criterion is utilized only if that criterion is failed as approved by function g(); while that is unsuccessful, the third, and every successive criteria are utilized.

Merge: By perception utilized to integrate a big amount of criteria taking all the criteria all at once with the intention of having the tests ordered. E.g., one recognition of merge can utilize an operator to incorporate the entire matrices into a single compound matrix, which is utilized to order the test cases.

This is done, e.g., by the operator Array flatten in mathematical amongst the respite of the operations, and horizontally unites two or more matrices in order to create a new matrix. In fact, the amount of rows in the entire matrices being united should be alike for an ensuing matrix.

Choice: By perception, the third as well as final ways of uniting the criteria selects one of a collection of criteria applying a selection function. It is unlike to Rank, each criterion contains the opportunity for choosing based upon the coverage criteria; as well, a formerly selected criterion might be chosen once more by the selection function. Here, the invocation function understands the choice utilizes each matrix (vector) stated in selection subsequently in diverse next () invocations; the diverse outcomes of these next() get logged. The "goodness" of every result is computed by further steps. The selection encompassing the utmost goodness and it is selected to be the outcome. For the respite of the unordered tests, the process is recurrent till all the tests turn out to be ordered.

Representing Higher Order Criteria

Rank, Choice, and Merge are combined for producing higher order condition. Mainly, the need for higher order criteria is an inspired one. Taking the sample of Merge with S and B; it is to be recalled that T_3 is chosen initially, after that a draw presented among T_2 and T_4; eventually, the respite of the tests are included in any series. The$Rank_{H(C_x,C_y,..C_Z)}$ invocation function is to be utilized with the intention of executing the higher order criterion. For testcase prioritization, Hybrid criteria function is used in the company of test cases function values that are enhanced applying the Adaptive Firefly Algorithm (AFA) Algorithm.

3.1.1. ADAPTIVE FIREFLY ALGORITHM (AFA)

So as to enhance problems, Firefly Algorithm (FA) mimics the bioluminescent communication behaviours. It presumes that the each and every fireflyis unisex and go in the space impacted by the flash bright of other fireflies. In order to idealize the flashing behaviour of fireflies in FA, three rules are created:

- Each and every fireflyis unisex and they are fascinated to other fireflies irrespective of their sex.
- The attraction of fireflies to others is relative to their brightness. Therefore fireflies containing minimum brightness are fascinated by the brighter ones as well as move towards them.
- To some optimization problem, the brightness of a firefly is identified or impacted by the fitness of the solution that the firefly provides in decision space.

In every generation, to enhance HTC, the fireflies move in the direction of the brighter neighbours as per Eq. (1)

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(rand - 0.5) \qquad (1)$$

The second term is the brightness attraction. γ is known as absorption coefficient. β_0 is known as greatest attractiveness value, it provides the attractiveness while $r_{ij} = 0$, r_{ij} is called the distance amid testsuite (firefly) i and testsuite j. It creates the attraction of a testsuite to additional is highest value while they are at the identical place when less while the distance is ∞ . The third term α is known as the randomization search and rand is called a random number produced as per uniform distribution [0, 1]. The three parameters are set in this manner: $\beta_0 = 1, \alpha \in (0,1)$and [0.01, 100] is suggested. Generally FA algorithm is shown to carryout superior to other algorithms; on the other hand it contains some weaknesses. Its parameters are pre-set and they perform well on functions with minimum amount of test suites. On the other hand, since the dimension as well as variable ranges rise, these $\beta_0 = 1, \alpha \in (0,1)$parameters must not be appropriate and the FA would get nastiest outcomes. Take the $\beta_0$attractiveness coefficient, its value $= \beta_0 e^{-\gamma r_{ij}^2}$. While handling the low dimension as well as small scale problems, the distance amid neighbor fireflies are minimum, the value of is inside a sensible range. On the other hand, while handling high-dimensional as well as significant problems, the distance r would be very large as well as the value of turn out to be extremely minimum. Second, the randomization term of Eq. (16) is not quite sensible. Consequently taking that the scales on diverse dimensions might change meaningfully, it is recommended to utilize $S_k\alpha$rather thanα. Here S_k is known as the scaling parameter denotes the ranges on all dimensions. A significant rule to develop a fine algorithm is to poise its exploration as well as exploitation capability, which assure the algorithm could search sensibly and converge to the optimal point. On the other hand, the randomization step in Eq. (1) is [−0.5, 0.5] with α = 1.

Adaptation

This research provides a novel function as the attractiveness coefficient excluding m self-adapt as per the test case HTC must be enhanced. Dependent upon the weakness as well as its cause of the algorithm, m must be the function of the test suite dimension as well as variable ranges, and these are called earlier resolving. Present m = K $\sqrt{Dim \times testsuiterange}$as the adaptive function. K is known as a constant number. Dim denotes the dimension of the HTC optimization and Range denotes the variable range on one dimension.

Decreasing Randomization

The randomization term of the FA algorithm would bright out excessive uneasiness in the late phase. Two parameters $\alpha_0$and α_{end} are presented and αreduces from $\alpha_0$and α_{end} linearly with iterations. The technique could maintain balance amid the exploration as well as exploitation capabilities of the afresh algorithm. In the initial stage, superiorαp gives healthier global searching of HTC as well as diversity. In the late stage, lesserα value evades the longstep jumping and provides improved convergence. For problems that scales change on diverse test suites, utilizing S_k multiply by the reducingα is as well recommended.

Position updating

Consequently, moving condition in AFA algorithm testsuite (firefly) i moves in the direction of testsuite (firefly) j while the fitness of the testsuite (firefly)j is superior to i and the jth testsuite's flash flag is correct.

The flash flags are not continued. For all fireflies, a counter would record the iterations of the testsuite endured its present state. As per a probability p computed by Eq (2),the flash flags change states. count$_i$ is known as the value of the ith firefly's state counter.

$$p_i = 0.5 + 0.1 \times count_i \quad (2)$$

Therefore the greater counts a firefly stays the state, the higher probability the state would changes. The highest count a firefly stay its present state is five. Subsequent to the state modifications, the value of equivalent counter would be reset to zero. This bionic technique creates that every firefly wink in a natural manner. In the origin FA algorithm, while population is created, the attraction movements are set up. On the other hand, in the AFA algorithm, randomness is included into the attraction term. While the best neighbor is not lighting now, a firefly moves in the direction of less-better neighbors. Consequently, this method mimics the winking behaviors could improve the diversity as well as avoid premature.

```
Pseudo code: Adaptive firefly algorithm
1. Objective function f(x), x=(x1,x2,…,xd)^T
2. Initialize a population of fireflies $x_i$ (i=1,2,…,n)
3. Define light absorption coefficient $\gamma$
4. While (t < MaxGeneration)
5. For i=1:n (all n fireflies)
6. For j=1:i
7. Light intensity is determined by f($x_i$)
8. If ($I_i > I_j$)
9. Move firefly i towards j in all d dimensions
10. Else
11. Move firefly i towards best solution in that iteration
12. End if
13. Attractiveness varies with distance via exp$^{(-\gamma r2)}$
14. End for j
15. End for i
16. Rank the fireflies and find the current best
17. Define normal distribution
18. For k = 1: n all n fireflies
19. Draw a random number from defined distribution
20. Evaluate new solution(new_cost(k))
21.If ((new_cost (k) < cost (i)) & & (new_cost(k)< last_cost_iteration(k)))
22. Move firefly i towards current best
24. End if
25. End for k
26. End while
27. Postprocess results and visualization
```

3.2. SOFTWARE EDC ESTIMATION

Test cases result is either true or false, it depends on the expected result and actual result.

If<expected results> equal to < Actual results>

Then <Test case result>should be <True>

Test case priority is given based on the severity of the Bug.

If <Test case>result=True<Priority> should be <Low>

else If<Test case>result=False && not affecting other module<Priority> should be <Medium>

else If<Test case>result=False && affecting other module <Priority>should be<High>

IV. RESULTS AND DISCUSSION

In this part, a numerical evaluation of the recommended methodology done in terms of different performance standards to examine performance enhancement of the projected and existing research methodologies. The java simulation setting is used to execute the proposed methodology. The execution measures studied in this work are listed as follows "Accuracy, Sensitivity, Specificity, Precision." The proposed research methodology implemented in the java simulation environment which evaluated for its performance over the performance metrics namely. The comparison made between the proposed method Regression Testing based Software EDC Estimation Technique (RTSEDCET)and the existing methodologies COCOMO.

Accuracy: It is defined as degree of correct identification of software project estimation. That is less false positive rate. The accuracy of the proposed system should be more than the other existing methodologies. The accuracy value is calculated in terms of the drug prediction system's true positive, false positive, true negative and false negative values. The accuracy is calculated as like as follows:

$$\text{Accuracy} = \frac{T_p}{(T_p + F_p + F_n)}$$

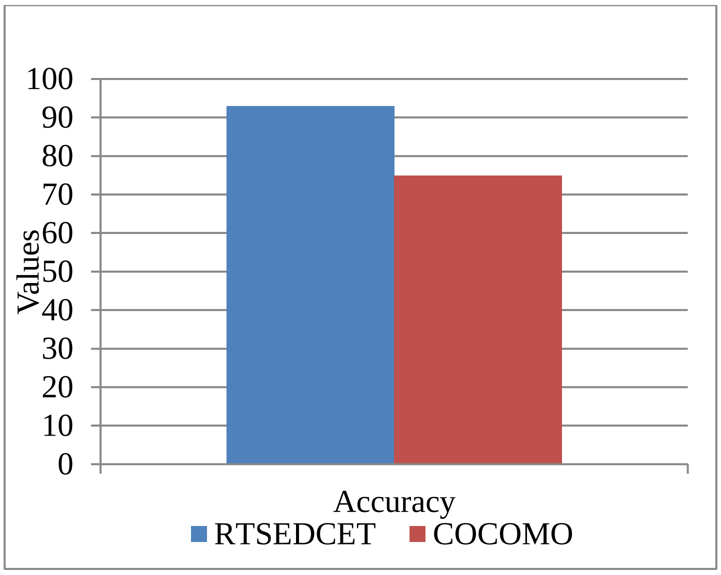

Figure 1. Accuracy comparison

The figure 1 illustrates the accuracy metric comparison graphical representation. This graphs shows that the proposed research method is better than the existing research methods. From this graph it is proved that the RTSEDCET is 22% better than COCOMO.

Sensitivity: It is also called as true positive rate which is defined as degree of correctly predicting the software project estimation. Mathematically, this can be expressed as:

$$\text{Sensitivity} = \frac{\text{number of true positives}}{\text{Number of true positives} + \text{Number of false negatives}}$$

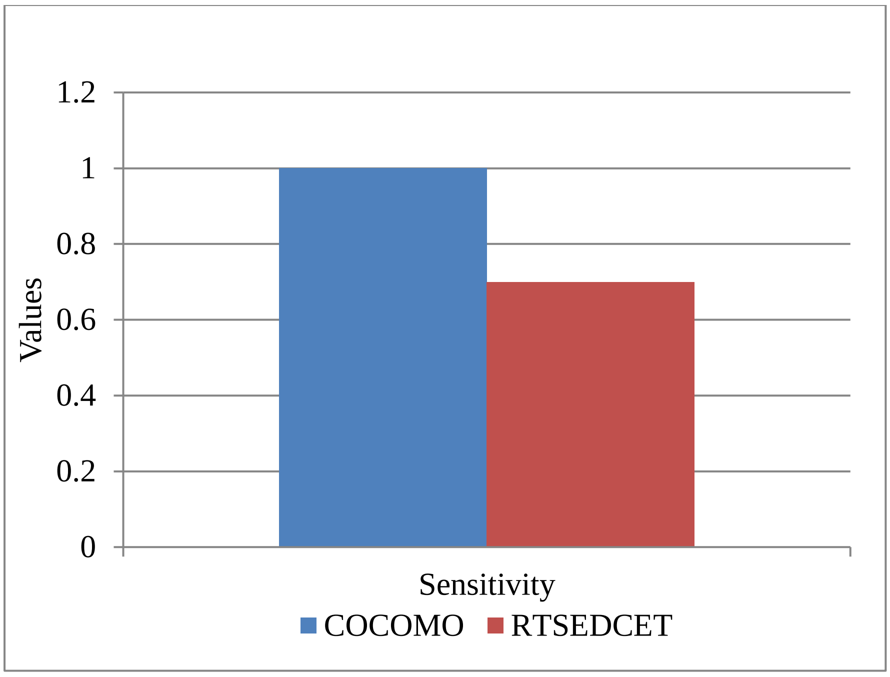

Figure 2. Sensitivity comparison

fig2 depicts sensitivity metric relating to graphical design. This graph reveals suggested approach is beneficial than existing research methods. From this graph it is proved that the RTSEDCET is 43% better performance than COCOMO.

Specificity: It is also called as true negative rate which is defined as degree of correctly predicting the software project estimation cost. Mathematically, this can also be written as:

$$\text{Specificity} = \frac{\text{number of true negatives}}{\text{Number of true negatives} + \text{Number of false positives}}$$

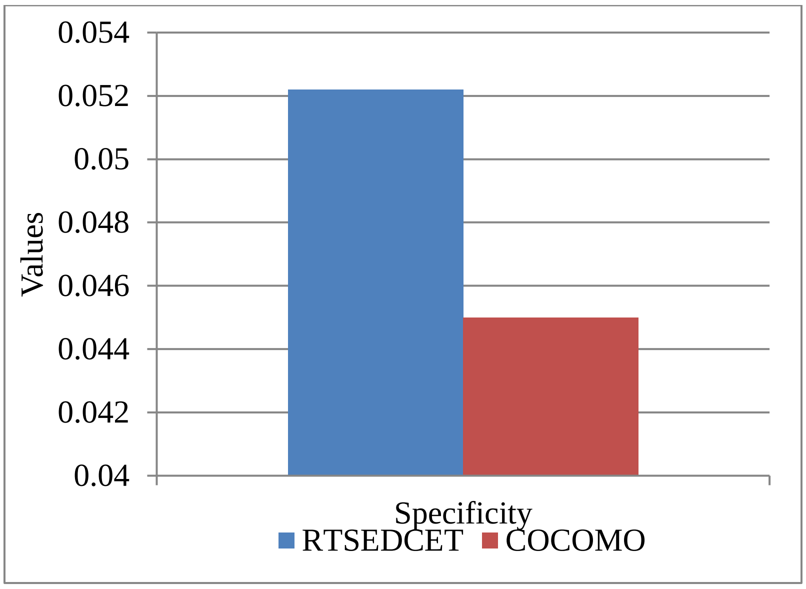

Figure 3. Specificity comparison

fig 3 demonstrates specificity metric measuring graphical illustration. This graph determines proposed method is helpful than existing research methods. From this graph it can proved that RTSEDCET ensures 16% better performance than COCOMO.

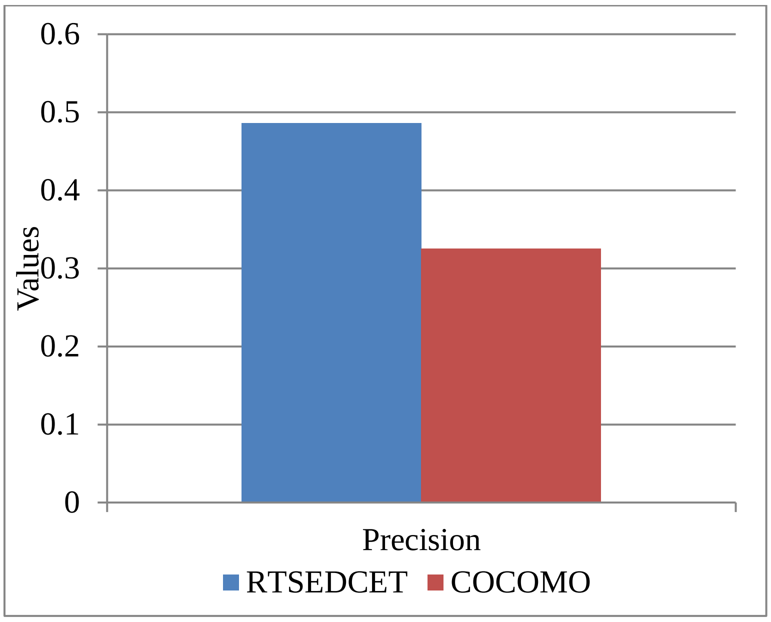

Figure 4. Precision comparison

Over fig Represents precision metric measuring graphical description. This chart explicates intended research process is helpful than current research methods. From this comparison it is proved that the RTSEDCET is 50% better performance thanCOCOMO.

IV. CONCLUSION

The experimental analysis is carried out on two datasets namely NASA 93 and COCOMO datasets. The proposed regressing testing model would generate the various test cases by comparing the attribute values of these datasets to predict the quality of the software in terms of effort, duration and cost. Based on these test case values, software project estimation can be done efficiently. In this work, adaptive firefly algorithm is utilized for the efficient test case generation which would combine the multiple attributes of the dataset to generation optimal test cases with the concern of ranking. The practical of the examination work conducted on java simulation context of which it explained that the proposed research technique leads to ensure the optimal consequence than existing examination techniques.

REFERENCE

1 Kerzner, H., & Kerzner, H. R. (2017). Project management: a systems approach to planning, scheduling, and controlling. John Wiley & Sons.
2 Trendowicz, A., & Jeffery, R. (2014). Software project effort estimation. Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags, 277-293.
3 Mohagheghi, P., & Aparicio, M. E. (2017). An industry experience report on managing product quality requirements in a large organization. Information and Software Technology, 88, 96-109.
4 Heagney, J. (2016). Fundamentals of project management. Amacom.
5 Nagaraj, S. H., Waddell, N., Madugundu, A. K., Wood, S., Jones, A., Mandyam, R. A., ... & Grimmond, S. M. (2015). PGTools: a software suite for proteogenomic data analysis and visualization. Journal of proteome research, 14(5), 2255-2266.
6 Grønbæk, K., Grudin, J., Bødker, S., & Bannon, L. (2017). Achieving cooperative system design: shifting from a product to a process focus. In Participatory Design (pp. 79-97). CRC Press.
7 Turk, D., France, R., & Rumpe, B. (2014). Limitations of agile software processes. arXiv preprint arXiv:1409.6600.
8 Cherkasova, L., Zhang, Q., Mathews, G. and Greene, W., Hewlett-Packard Development Co LP, 2015. Capacity planning for computing systems hosting multi-tier application based on think time value and resource cost of composite transaction using statistical regression analysis. U.S. Patent 9,135,075.
9 Weigend, A. S. (2018). Time series prediction: forecasting the future and understanding the past. Routledge.
10 Gkioxari, G., Toshev, A., & Jaitly, N. (2016, October). Chained predictions using convolutional neural networks. In European Conference on Computer Vision (pp. 728-743). Springer, Cham.
11 Petitpierre, B., Broennimann, O., Kueffer, C., Daehler, C., & Guisan, A. (2017). Selecting predictors to maximize the transferability of species distribution models: lessons from cross-continental plant invasions. Global Ecology and Biogeography, 26(3), 275-287.
12 Suri, P. K., & Ranjan, P. (2012). Comparative analysis of software effort estimation techniques. International Journal of Computer Applications (0975–8887), 48(21).
13 Patil, L. V., Waghmode, R. M., Joshi, S. D., & Khanna, V. (2014, February). Generic model of software cost estimation: A hybrid approach. In Advance Computing Conference (IACC), 2014 IEEE International (pp. 1379-1384). IEEE.
14 Waghmode, R. M., Patil, L. V., & Joshi, S. D. (2013). A Collective Study of PCA and Neural Network based on COCOMO for Software Cost Estimation. International Journal of Computer Applications, 74(16).
15 Madheswaran, M., & Sivakumar, D. (2014, July). Enhancement of prediction accuracy in COCOMO model for software project using neural network. In Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on (pp. 1-5). IEEE.

16 Saroha, M., & Sahu, S. (2015, May). Tools & methods for software effort estimation using use case points model—A review. In Computing, Communication & Automation (ICCCA), 2015 International Conference on (pp. 874-879). IEEE.

17 Kaushik, A., Chauhan, A., Mittal, D., & Gupta, S. (2012). COCOMO estimates using neural networks. International Journal of Intelligent Systems and Applications, 4(9), 22.

18 Gupta, A., Mishra, N., & Kushwaha, D. S. (2014, February). Rule based test case reduction technique using decision table. In Advance Computing Conference (IACC), 2014 IEEE International (pp. 1398-1405). IEEE.