Software Quality Evaluation by Cocomo II With NN and SVM

Naveen Malik#1, Sandip Kumar Goyal#2
#1Ph.D Scholar, CSE, MM Deemed To be University Mullana,Ambala,India
#2Professor, CSE, MM Deemed To be University Mullana,Ambala,India
Email-naveenmalik317@gmail.com
Email- skgmmec@gmail.com

Abstract
Cost, time and quality projection are the crucial aspects in software development process. Incorrect estimations can cause losses which in turn may lead to irreversible damage. It is generally perceived that a imperfectly estimated project always results in a substandard quality due to the efforts being wrongly directed. Firstly Effort Estimation is calculated by actual effort and proposed Effort. That Effort evaluation of 500 NASA projects, after that evaluation is done by four parameters Standard Error, Standard Deviation, Mean Absolute Error, Root Mean Square Error. The author amalgamated the robustness of COCOMO-II with that of Neural Network NN and Support Vector Machine SVM .Quality Which we evaluate that is quality Evaluation of Semantic Web Application. In the last checks the majority of all four parameters for software quality assessment.

Keywords: Software quality, Neural Network, SVM, COCOMO II, Semantic Web.

I. INTRODUCTION
Software development process involves many attributes such as testability, effort estimation, accuracy and usability etc. The effort estimation grabs our attention very easily. The current era is bestowed with a lot of technical developments that leads to competition amongst everything born out of these developments. This paves the way for effort estimation to proffer extremely vigorous results with high facets of reliability and accuracy before kicking off the project [1, 2]. The evolution in software project is gingerly and a definite forecast is not viable in factual sense [3, 4]. It is foremost to recognize the issues regarding such type of prognosis that might wind up in overestimation or underestimation for the efforts.

Estimation the effort is multifarious task and the estimation prototypes handling such jobs are categorized as algorithm grounded COCOMO model, non-algorithm grounded trained model and also the models that take on the robustness of machine learning architectonics [5, 6]. Countless effort estimation prototypes developed with the thriving necessity but these could not touch the perfection ceiling yet. PRICE-S model by Park [7], COCOMO by Boehm [8], Putnam and Myers SLIM [9] and Function Point developed by Albrecht [10] hold the lion’s share amongst the algorithmic prototypes.

The models pose some restraints in carrying off the faultless estimation due to the requirement of input attributes in view of Line Of Code, complexities that could not be squarely attained at preliminary steps of development process. This induces the incapacity of the models to put forward solutions to resist composite relationships, graded data along with intense deficiency in interpretation potentiality [11]. In this study,
COCOMO is utilized in integration with artificial bee colony algorithm to provide more vigorous effort approximation in respect of both sensitivity and accuracy. Furthermore, the paper is organized in the following segments:

Segment I: Sets forth the quality evaluation and the bit part of effort approximation in quality evaluation
Segment II: Talks over the COCOMO substructure
Segment III: Narrates the proposed work procedure and the algorithm employed
Segment IV: Mentions the formula and evaluation attributes
Segment V: Brings up the results
Segment VI: Winds up the paper
Segment VII: Quotes the reference work

II. QUALITY EVALUATION AND EFFORT ESTIMATION
The quality of software is dependent on numerous facets inclusive of the used effort in the route of software development. Productivity is related to the skill of the people deployed to the correct product. Productivity will be lesser and efforts will be higher when the people are not skill oriented. Researchers have always been interested in the quality evaluation and this work intends to classify the quality on the basis of effort approximation.

Fig. 1. Software objects count
III. COCOMO SUBSTRUCTURE
Boehm developed the COCOMO. It provides algorithm induced cost approximation. Regression formula is the base for the estimation operability in this model. Feature and historical data of the active project lead to input attributes for the model. The model operates in three modes:
- Organic mode
- Semi-detached mode
- Embedded mode

Organic mode revolves around the working of simple projects with teams working in a well explanatory and firm situations. The teams involved in semi-detached mode manifest diverse knowledge. Embedded mode employs strict checks to match the differing requirements. The basic effort is approximated as follows:

\[ E_{\text{months}} = p_1 \times (KLOC) p_2 \]  \hspace{1cm} (1)

A. COCOMO-II PROTOTYPE

Fig. 2. COCOMO II Prototype

COCOMO-II is a mathematical and idolized solution to evaluate the effort on a project. The following mathematical parameters have been utilized as:

i. Required Software Reliability (RELY)
   It’s an evaluation of the threshold of the extent up to which software must perform

ii. Data Base Size Data
   It is a measure of data requirement for the proceeding

iii. Product Complexity (CPLX)

The product complexity is divided into five sections:
   a) Control Operations CPLX
   b) Computation Operation CPLX
   c) Device Dependent CPLX
d) Data Management CPLX

e) User Interface CPLX

iv. Required Reusability (RUSE)
   This is the required extra effort to complete the project.

v. Execution Time (TIME)
   Total execution time of the project is calculated by the execution time

vi. Storage Constraints (STOR)
   Total amount of storage required for the project can be termed as shown

vii. Programmer Capability (PCAP)
   It is a parameter which is affected by the programmer’s capability. It is influenced by the way a programmer’s capability.

viii. Language and Tool Experience (LTEX)

Regression Analysis is the basis of this estimation model. The prototype manifests the architecture as mentioned below [12-13]:

- Application Composition model functions on the speculation that the reclaimable attributes such as scripts and record encodings blueprint the fundamental architecture of any system. It is based on the consideration of the sample efforts to cope with issues concerning user interface, performance, software and system relationship. Effort estimation is accomplished in the initial stages and the size is estimated on the basis of application or the object points such as screens and project reports etc.
- Initial Design Model focuses on the forecast of the duration and the intricate cost of the project earlier than the complete design could. Size is calculated by utilizing the unadjusted function points in conjunction with prediction equations.
- Post Architecture Prototype revolves around the bonafide design and prolongation of the software to accomplish the correct prediction of the size of the product. This model is cost efficacious when system risk, task and perceptions are under consideration. LOC or the function points are used for size forecast.

IV. PROPOSED WORK

The proposed work used the data set of 500 projects in the NASA data sets. In the proposed model, COCOMO-II received the feed from ABC in the form of attributive values. Moreover, NN was used for classification amongst the values before being fed to COCOMO-II. Support parameter is accessed by Effort. There is no rule if there is high effort that means quality is high or if there is less effort that means there is no quality low of that software.

In the below table we prescribed the range of 4 cost drivers which are used in our proposed model. 18 cost drivers used in post architecture model. These cost drivers are compatible with Reliability, Programmers capability, Required Reusability, Language and Tool experience of the architecture model. All the Effort Estimation And Quality Evaluation process have been done in MATLAB Machine. Firstly Effort Difference Estimation is find by the difference of Actual Effort and Proposed Effort. In the last We apply the Rule Set for Quality Evaluation.
V. RESULTS

In this section comparison of the effort estimation obtained using proposed model is compared with the actual effort and the effort obtained from COCOMO-II and Neural Network and SVM [16]. Effort is estimated of 500 ISRO projects, firstly we evaluate 100 software quality and take their mean value, Same for next 100 software, difference of actual effort and proposed effort is effort estimation.

Below all these tables quality evaluation by four different parameters Standard Error, Standard Deviation, Mean Absolute Error, Root Mean Square Error. All four parameters evaluate different quality evaluation. All mean values are taking of 100-100 data sets for calculating their mean values. In the Below Tables Tp Neural is denoted by True Positive of Neural, Tp SVM is denoted by True positive SVM, Tp COCOMO II is denoted by True Positive of COCOMO II Model. Same as Tp Phenomenon we used Fp For False positive for all models. After the effort estimation on MATLAB, we find quality evaluation by four parameters SE, SD, MAE, RMSE. In the last we find out accuracy of particular software by the Recall and Precision of COCOMO II, NN, SVM. We apply the Rule Set

<table>
<thead>
<tr>
<th>Number of Project files</th>
<th>tp Neural</th>
<th>tp SVM</th>
<th>tp COCOMO 2</th>
<th>fp Neural</th>
<th>fp SVM</th>
<th>fp COCOMO2</th>
<th>fp Proposed</th>
<th>fp Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.74</td>
<td>0.6798</td>
<td>0.61452</td>
<td>0.26</td>
<td>0.3202</td>
<td>0.364545</td>
<td>0.7896</td>
<td>0.10256</td>
</tr>
<tr>
<td>200</td>
<td>0.7321</td>
<td>0.6685</td>
<td>0.62348</td>
<td>0.2679</td>
<td>0.3315</td>
<td>0.37652</td>
<td>0.745512</td>
<td>0.107741</td>
</tr>
<tr>
<td>300</td>
<td>0.77145</td>
<td>0.6785</td>
<td>0.66324</td>
<td>0.22855</td>
<td>0.3215</td>
<td>0.33676</td>
<td>0.7965</td>
<td>0.108963</td>
</tr>
<tr>
<td>400</td>
<td>0.72145</td>
<td>0.69325</td>
<td>0.701458</td>
<td>0.27855</td>
<td>0.30675</td>
<td>0.298542</td>
<td>0.78552</td>
<td>0.1569</td>
</tr>
<tr>
<td>500</td>
<td>0.71456</td>
<td>0.699741</td>
<td>0.700145</td>
<td>0.28544</td>
<td>0.300259</td>
<td>0.299855</td>
<td>0.75689</td>
<td>0.14125</td>
</tr>
</tbody>
</table>
**TABLE II.**

<table>
<thead>
<tr>
<th>Number of Project files</th>
<th>Precision Neural</th>
<th>Precision SVM</th>
<th>Precision COCOMO2</th>
<th>Recall Neural</th>
<th>Recall SVM</th>
<th>Recall COCOMO2</th>
<th>Precision Proposed</th>
<th>Recall Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.74</td>
<td>0.6798</td>
<td>0.627660063</td>
<td>0.87634116</td>
<td>0.771623156</td>
<td>0.719477357</td>
<td>0.885043</td>
<td>0.88539</td>
</tr>
<tr>
<td>200</td>
<td>0.7321</td>
<td>0.6685</td>
<td>0.62348</td>
<td>0.87414925</td>
<td>0.765750286</td>
<td>0.721052875</td>
<td>0.873729</td>
<td>0.872748</td>
</tr>
<tr>
<td>300</td>
<td>0.77145</td>
<td>0.6785</td>
<td>0.66324</td>
<td>0.86190716</td>
<td>0.746014294</td>
<td>0.76794998</td>
<td>0.87966</td>
<td>0.883018</td>
</tr>
<tr>
<td>400</td>
<td>0.72145</td>
<td>0.69325</td>
<td>0.701458</td>
<td>0.87068339</td>
<td>0.748367248</td>
<td>0.748574795</td>
<td>0.833514</td>
<td>0.884515</td>
</tr>
<tr>
<td>500</td>
<td>0.71456</td>
<td>0.699741</td>
<td>0.700145</td>
<td>0.85253412</td>
<td>0.765756065</td>
<td>0.773610965</td>
<td>0.842731</td>
<td>0.881068</td>
</tr>
</tbody>
</table>

**Fig. 4. Precision Values**

**Fig. 5. Recall Values**
Fig. 6. F-Measure Values

Fig. 7. Accuracy
TABLE IV.

<table>
<thead>
<tr>
<th>Number of Project files</th>
<th>f-measure Neural</th>
<th>f-measure SVM</th>
<th>f-measure COCOMO2</th>
<th>f-measure Proposed</th>
<th>Accuracy Neural</th>
<th>Accuracy Proposed</th>
<th>accuracy SVM</th>
<th>Accuracy COCOMO2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.8024203</td>
<td>0.722807018</td>
<td>0.670439699</td>
<td>0.88521668</td>
<td>56.27718797</td>
<td>70.79265</td>
<td>47.45218</td>
<td>41.15691</td>
</tr>
<tr>
<td>200</td>
<td>0.7968435</td>
<td>0.713828083</td>
<td>0.668726001</td>
<td>0.8732384</td>
<td>55.2299879</td>
<td>69.90051</td>
<td>47.24882</td>
<td>41.38297</td>
</tr>
<tr>
<td>300</td>
<td>0.8141738</td>
<td>0.71065724</td>
<td>0.711764548</td>
<td>0.88133609</td>
<td>58.67700078</td>
<td>70.96243</td>
<td>47.26675</td>
<td>45.419</td>
</tr>
<tr>
<td>400</td>
<td>0.7890727</td>
<td>0.719754977</td>
<td>0.7242509</td>
<td>0.85825731</td>
<td>59.63914782</td>
<td>68.16442</td>
<td>48.15407</td>
<td>47.50012</td>
</tr>
<tr>
<td>500</td>
<td>0.7774731</td>
<td>0.731261669</td>
<td>0.735046863</td>
<td>0.8614728</td>
<td>62.7831374</td>
<td>68.38358</td>
<td>49.13635</td>
<td>48.01631</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

Over the time, efforts and quality estimation are being done by various authors to overcome the degradation of the developed software. It is really difficult to obtain a satisfactory prediction well before the starting of the project as on the way project gets evolved and influence by numerous factors on the go. In the present work, authors tried to reach a near ideal effort estimation with the combination of ABC algorithm to search for the produced quality at the end. The solution obtained from ABC is then fed to NN for classification and training to enhance the quality of results. Finally, COCOMO-II is employed to predict the effort estimation parameters. After our proposed work we can says that if any Project quality is degraded on the basis of SE,SD, MAE, RMSE that means quality of software is degraded. We checks the majority of all parameters that is shown in below graphs of F-measure and Accuracy Measure.

![Fig. 8.F-Measure](image-url)
References