Chapter 4

INTERMEDIATE STAGE ESTIMATION

In any process model with respect to software development, during intermediate stage, the development team has clear understanding of client’s requirements. More details of project are available. Higher estimation accuracy is expected during these stages. Primary focus towards estimates is to optimize resource utilization, whereas, during initial stage costing is prime focus. Early Design Model and Post Architecture Model of COCOMO II are applicable at this stage. Assumption is made that analysis & design models are available. UML diagram such as Use Case diagram with Cockburn Templates are prepared and Class Diagrams, Component Diagrams are ready for refinement. It is revealed from study that Artificial Neural Network, Fuzzy Logic based AI models are applicable. Assuming that there is no replacement for human expert estimator, the estimation accuracy with AI technologies can be improved. The justifying points are as mentioned below

- Neural networks are low-level computational structures that perform well when dealing with raw data
- Fuzzy logic deals with reasoning on a higher level, using linguistic information acquired from domain experts. It can process qualitative data in the form of vague knowledge.
- However, fuzzy systems lack the ability to learn and cannot adjust themselves to a new environment.
- Neural networks can learn, but they are opaque to the user.

These AI technologies are evolved as result of attempts made for enabling computer to do things which people do better. Human can learn and process vague information, data and
knowledge as well as apply in different situation. The Neuro-Fuzzy inference system exactly mimic the same features of human up to limited extent in restricted domain.

When new software project is in hand, effort estimates are required to derive with vague requirements and there is uncertainty about the resource quality and availability. Still human expert estimate and manage software development project and complete it.

COCOMO II, is widely accepted algorithmic estimation model, which deals with historical data.

Cost drivers which includes effort multipliers and scale factors are represented in qualitative terms such as Very Low, Very High etc. These qualitative terms are replaced by quantities derived from historical data of projects and are used in equation to calculate estimates such as effort in man-months or schedule in months as indicated in Table 2.4 & 2.5.

It is revealed from literature that, COCOMO II model is found suitable to apply Adaptive Neuro-Fuzzy Inference System to improve its accuracy and easy to use for novice estimator [41][43][71].

4.1 ADAPTIVE NEURO FUZZY INFERENCE SYSTEM (ANFIS)

The ANFIS is a FIS implemented in the framework of an adaptive fuzzy neural network. A typical architecture of ANFIS is depicted in the Figure 4.1, in which a circle indicates a fixed node, whereas a square indicates an adaptive node. For simplicity, it was assumed that the FIS as two inputs $x$ and $y$ and one output $z$ [44].

The ANFIS implements a first-order Sugeno fuzzy model. For this model, a typical rule set with two fuzzy if-then rules is expressed as

**Rule 1:** If $x$ is $A_1$ and $y$ is $B_1$, then $z_1 = p_1x + q_1y + r_1$.

**Rule 2:** If $x$ is $A_2$ and $y$ is $B_2$, then $z_2 = p_2x + q_2y + r_2$. 
where $A_i$ and $B_i$ are the fuzzy sets in the antecedent, and $p_i$, $q_i$ and $r_i$ are the design
parameters that are determined during the training process.

The system used is a hybrid intelligent system which combines two intelligent technologies
viz. neural networks with a fuzzy inference system resulting in a hybrid Neuro-fuzzy system.

When a representative set of examples is available, a Neuro-fuzzy system can automatically
transform it into a robust set of fuzzy IF-THEN rules, and thereby reduce our dependency on
expert knowledge while building intelligent systems.

**Figure 4.1 Adaptive Neuro-fuzzy Inference System**

**Layer 1:** Every node $i$ in this layer is a square node with a node function

$$O_i^1 = \mu_{A_i}(x),$$
Where $x$ is the input to node $i$, $A_i$ is the linguistic label associated with this node function. That is to say, $O_i$ is the membership function of $A_i$ and it specifies the degree to which the given $x$ satisfies the quantifier $A_i$. Superscript 1 of $O_i$ indicates this is the output of first layer. Parameters in this layer are referred to as the premise parameters.

**Layer 2:** Every node in this layer is circle node labeled II, which multiplies the incoming signals and sends the product out. Each node output represents the firing strength (or weight) of a rule.

**Layer 3:** Every node in this layer is a circle node labeled N. The $i$-th node calculates the ratio of the $i$-th rule’s firing strength to the sum of all rules’ firing strengths.

**Layer 4:** Every node $i$ in this layer is a square node with a node function,

$$O_i^4 = w_i(p_i x + q_i y + r_i),$$

where $w_i$ is the output of layer 3, and $\{p_i, q_i, r_i\}$ is the parameter set. Parameters in this layer are referred to as consequent parameters.

**Layer 5:** It is a circle node that sums all incoming signals.

It is clear that the ANFIS has two set of adjustable parameters, namely the premise and consequent parameters. During the learning process, the premise parameters in the layer 1 and the consequent parameters in the layer 4 are tuned until the desired response of the FIS is achieved. The hybrid learning algorithm, which combines the least square method and the back-propagation algorithm, is used to rapidly train and adapt the FIS. When the premise parameter values of the Membership Functions are fixed, the output of the ANFIS is represented as a linear combination of the consequent parameters.
The Least Square Method is used to determine optimally the values of the consequent parameters. When the premise parameters are not fixed, the search space becomes larger and the convergence of training becomes slower. The hybrid learning algorithm can be used to solve this problem. This algorithm has a two-step process. First, while holding the premise parameters fixed, the functional signals are propagated forward to layer 4, where the consequent parameters are identified by the Least Square Method. Then, the consequent parameters are held fixed while the error signals, the derivative of the error measure with respect to each node output, are propagated from the output end to the input end. The premise parameters gets updated by using Back-propagation algorithm.

4.2 COCOMO II : COST DRIVERS FUZZY INPUT
Algorithmic model, COCOMO II accept size, constants and cost drivers as input to calculate Effort in person-months (COCOMO II 2000). The following equations are used for both Early Design Model and Post Architecture Model.

\[(5.1)\]

Where:

\[\begin{align*}
PM & \quad \text{Person Months of estimated effort} \\
A & \quad \text{Constant set as 2.5} \\
BRAK & \quad \text{Breakage : Percentage of code thrown away due to Requirement volatility} \\
N & \quad \text{7 for Early Design Model & 17 for Post Archi.Model} \\
SF & \quad \text{Scale Factors : PREC, FLEX, RESL, TEAM, PMAT} \\
EM & \quad \text{Effort Multipliers : RCPX, RUSE, PDIF, PERS, PREX, FCIL, SCED}
\end{align*}\]
The exponent B is an aggregation of five scale factors viz. Precededness (PREC), Development Flexibility (FLEX), Architecture / Risk Resolution (RESL), Team Cohesion(TEAM), Process Maturity(PMAT) and their values are in the range from Very Low, Low, Nominal, High, Very High and Extra High.

The multiplication of EMs (Effort Multipliers) is part of equation. The number of values are depend upon the model. The Table 4.1 lists the effort multipliers in both models.

**Table 4.1 List Effort Multipliers in COCOMO II Models (COCOMO II,2000)**

<table>
<thead>
<tr>
<th>Early Design Cost Drivers</th>
<th>Combined Post-Architecture Cost Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERS (Personnel Capability)</td>
<td>ACAP (Analyst Capability), PCAP (Programmer Capability), PCON (Personnel Continuity)</td>
</tr>
<tr>
<td>RCPX (Product Reliability and complexity)</td>
<td>RELY (Required Software Reliability), DATA (Database Size), CPLX, (Product Complexity), DOCU (Documentation match to life-cycle needs)</td>
</tr>
<tr>
<td>RUSE (Required Reuse)</td>
<td>RUSE (Required Reusability)</td>
</tr>
<tr>
<td>PDIF (Platform Difficulty)</td>
<td>TIME (Execution Time Constraint), STOR (Main Storage Constraint), PVOL (Platform Volatility)</td>
</tr>
<tr>
<td>PREX (Personnel Experience)</td>
<td>AEXP (Application Experience), PEXP (Platform Experience), LTEX (Language and Tool Experience)</td>
</tr>
<tr>
<td>FCIL (Facilities)</td>
<td>TOOL (Use of Software Tools), SITE (Multisite Development)</td>
</tr>
<tr>
<td>SCED (Schedule)</td>
<td>SCED (Required Development Schedule)</td>
</tr>
</tbody>
</table>
4.3 MODIFIED ANFIS WITH COCOMO II FOR SOFTWARE ESTIMATION

In COCOMO II, each cost driver, either Scale Factor or Effort Multiplier has been considered as fuzzy variable and takes the discrete fuzzy value from term set. The Neuro-Fuzzy effort estimator model accept qualitative inputs as discrete fuzzy values. Fuzzy values are easy to understand for user as they are divided into six linguistic such as Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH) and Extra High (XH). The user has to represent cost driver coarsely in turn which is represented by single crisp value as indicated in Appendix B. These are calibrated values with historical data of projects. Scale Factor and Effort Multiplier fuzzy values are mapped into crisp values [41][47].

The coarse selection of input linguistic value for cost drivers has been identified as major source of inaccuracy in calculating estimates. Hence, the user who is expert / novice estimator or project manager, must be allowed to input finely. Hence, modified GUI has been proposed. Additional slider interface is provided to input cost driver. Estimator need not to worry about quantitative mapping.

For each linguistic variable universe of discourse is changed from discrete to continuous. The sliders can help the user to set any values from VL to XH on continuous scale instead of selecting discrete fuzzy values. A hint is also provided to the user in the form of question, which can be referred in times of confusion. When the user selects a value between any two linguistic terms (for e.g. Slider bar is between VL and L), relationship of the particular cost driver for both the linguistic terms and rules are fired accordingly with appropriate membership values for respective fuzzy set. It looks like that user is entering crisp value on continuous universe of discourse without knowing unifying relationship and fuzzyfication policy.

The model presented in Figure 4.2 uses ANFIS, trained with existing knowledgebase of the past projects, which has fuzzy values as well as calibrated numeric values for scale factors and effort multipliers. Input from estimator is accepted through questionnaire. Where user is
allowed to set each cost driver fuzzy value on continuous scale through slider. These are interpreted further. Each cost driver input is represented by at the most two linguistic terms with respective membership value of fuzzy set associated with each linguistic term. ANFIS which is trained with knowledgebase converts it into numeric value for each cost driver. Further they participate in equation of the Algorithmic Model COCOMO II to estimate effort.

The additional interface is developed to facilitate estimator to select intermediate fuzzy value in the range between two fuzzy terms. This system also reduces timely calibration for Fuzzy to Crisp mapping as projects added to knowledge-base.

Figure 4.2  AI Model for Software Estimation using ANFIS with COCOMO II
THE EXPERIMENT: ADVANCE COST ESTIMATOR

The experiment has been conducted to evaluate neuro-fuzzy approach for software estimation. Advance Cost Estimator, mainly effort estimation tool has been developed on the basis of neuro-fuzzy approach proposed in the paper[41]. The attempt has been made to overcome limitations of neuro-fuzzy approach with COCOMO. The experiment is carried with 63 projects data from COCOMO 81 and 35 projects data from NASA.

The data is collected from NASA and COCOMO literature (Appendix B) is used for the experiments. In a row for each project linguistic values for cost drivers are presented and last column indicates actual effort required for the project in person month. The ANFIS is trained with these data sets separately. It is suggested that software development enterprise has to append their completed project data and ANFIS is trained again.
Figure 4.4 COCOMO GUI to enter Cost Driver Qualitative Discrete Values

Figure 4.5 Modified GUI to enter Cost Drivers values on continuous scale with sliders
4.5 RESULTS: ANFIS WITH COCOMO II

The Neuro-Fuzzy model is evaluated by implementing and modifying interface for fuzzy input on continuous scale. The results are compared between ANFIS, Neural Network and COCOMO. Data available with COCOMO and NASA literature is used for experiment. Out of 63 projects, 53 projects data is used for training the ANFIS and 10 project data is used for testing. The result of effort estimation is depicted in Table 4.2.

As presented in Table 4.3, Mean Absolute Error, Mean Square Error and Root Mean Square Deviation for ANFIS-COCOMO-II with modified GUI is minimum as compared to other models such as COSTAR, COCOMO II and ANFIS with COCOMO II.

It is evident from the result that with this modification in GUI, accuracy of estimates is increased.

<table>
<thead>
<tr>
<th>Project Name</th>
<th>63 Project Database Reference</th>
<th>COCOMO</th>
<th>COSTAR</th>
<th>Actual</th>
<th>ANFIS without Slider</th>
<th>ANFIS with Slider</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>21</td>
<td>2059.314</td>
<td>1855.270</td>
<td>2455</td>
<td>2042.580</td>
<td>2349.060</td>
</tr>
<tr>
<td>p2</td>
<td>29</td>
<td>7.574</td>
<td>7.574</td>
<td>7.5</td>
<td>15.440</td>
<td>7.180</td>
</tr>
<tr>
<td>p3</td>
<td>33</td>
<td>536.582</td>
<td>619.708</td>
<td>605</td>
<td>466.490</td>
<td>451.900</td>
</tr>
<tr>
<td>p4</td>
<td>35</td>
<td>142.014</td>
<td>142.012</td>
<td>82</td>
<td>63.480</td>
<td>58.490</td>
</tr>
<tr>
<td>p5</td>
<td>37</td>
<td>38.064</td>
<td>43.956</td>
<td>47</td>
<td>46.020</td>
<td>46.020</td>
</tr>
<tr>
<td>p6</td>
<td>41</td>
<td>4.062</td>
<td>4.692</td>
<td>6</td>
<td>5.420</td>
<td>5.420</td>
</tr>
<tr>
<td>p7</td>
<td>42</td>
<td>48.070</td>
<td>48.068</td>
<td>45</td>
<td>45.330</td>
<td>47.370</td>
</tr>
<tr>
<td>p8</td>
<td>44</td>
<td>110.329</td>
<td>110.325</td>
<td>87</td>
<td>75.700</td>
<td>80.760</td>
</tr>
<tr>
<td>p9</td>
<td>53</td>
<td>22.419</td>
<td>22.420</td>
<td>14</td>
<td>16.500</td>
<td>16.500</td>
</tr>
<tr>
<td>p10</td>
<td>56</td>
<td>536.657</td>
<td>528.882</td>
<td>958</td>
<td>955.230</td>
<td>955.250</td>
</tr>
</tbody>
</table>
Table 4.3 Comparison of errors with different Estimation Models

<table>
<thead>
<tr>
<th>Error Analysis</th>
<th>ANFIS With Slider</th>
<th>COSTAR</th>
<th>ANFIS without Slider</th>
<th>COCOMO II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Absolute Error (MAE)</td>
<td>29.829</td>
<td>114.281</td>
<td>59.585</td>
<td>99.123</td>
</tr>
<tr>
<td>Mean Square Error (MSE)</td>
<td>3527.538</td>
<td>54827.144</td>
<td>18982.433</td>
<td>34308.820</td>
</tr>
<tr>
<td>Root Mean Square Deviation (RMSD)</td>
<td>59.393</td>
<td>234.152</td>
<td>137.777</td>
<td>185.226</td>
</tr>
</tbody>
</table>

Figure 4.6 Graph showing project-wise ln(estimated effort) by different models/methods compared with ln(actual effort)
The graph shown in Figure 4.6 clearly indicates that effort estimated with modified GUI are nearly equal to actual effort of respective projects.

4.6 THE EXPERIMENT WITH COMBINED METHODS

Three projects viz. P1, P2 and P3 have been selected for the experiment, for which sufficient documentation was available. Initially Request for Proposal document is referred. Each RFP was evaluated for completeness. Size in terms of Function points was derived for each. Then usecase diagram along with usecase text are studied and usecase point are computed. The analysis and design class diagrams were referred and class points are computed [8].

The experiment has been conducted with the consideration of RUP as a process model. The estimates are calculated during inception, elaboration phases and mapped as budgetary, initial and progressive estimates.

Budgetary estimates are calculated based on input given by experts from industry. They were asked to apply their knowledge and experience and suggest effort in terms of person-months with justification. The estimate with minimum error with all inputs is selected for each project. This method is synonymous to Expert Judgment Method.

Initial estimates are calculated by Function point method, COCOMO II Early Design Model and Usecase point method. Each Usecase point is weighed appropriately to calculate effort in Person-Months.

Progressive estimates are calculated by following COCOMO II Post Architecture Model and Class point method.

Since the effort estimation is not guaranteed for accuracy by any method. One of the approach is to apply more than one method and calculate effort estimate at different stage of software development process model. The experiment is conducted for evaluating effort estimates by applying contemporary sizing methods.
It is clearly evident from the values presented in Table 4.4, that the difference between estimated effort are converging. In absence of actual effort values, use of multiple estimation methods at any time during project life cycle controls drastic drifting with previous estimates.

**Table 4.4 Effort estimation at different stages combining various methods**

<table>
<thead>
<tr>
<th>Project-Budgetary Estimate</th>
<th>Initial Estimate</th>
<th>Progressive Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function Point</td>
<td>Early Design</td>
</tr>
<tr>
<td>P1-15</td>
<td>11.6</td>
<td>13.1</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>P2-30</td>
<td>25.6</td>
<td>29.8</td>
</tr>
<tr>
<td></td>
<td>27.4</td>
<td></td>
</tr>
<tr>
<td>P3-18</td>
<td>13.7</td>
<td>17.4</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Further this deviation with respect to actual can be minimized by combining these methods with AI models discussed in this dissertation. Semantic Matching of Software Project Documents can be applied while Budgetary Estimates. ANFIS can be combined with COCOMO II Early Design Model and Post Architecture Model.
Figure 4.7 Trend of calculated estimates at different stages

The Figure 4.7 clearly indicates that effort values at different stages are converging.

4.7 SUMMARY

This chapter presents evaluation of Adaptive Neuro-Fuzzy Inference System in combination with COCOMO II. The modification in GUI enabled estimator to enter cost driver values on continuous scale. The result of comparison indicated that Mean Absolute Error and Root Mean Square Error is reduced compared to other models.

Finally combination of different sizing approach are discussed and experimental result indicated that combination of more than one sizing approach and estimation methods/models reduces drifting from actual effort. The proposal is made for proper combination of different estimation methods/models at initial and intermediate stage estimation.

ANFIS greatly improved the accuracy of COCOMO II models for software estimation. Combined estimates are used as a reference to decide whether the estimates derived are converging to actual or not. If result of some method falls out of the trend, decision can be taken to omit the result or apply another method.