CHAPTER 2

LITERATURE REVIEW

For the estimation of Software cost, several researches have been suggested by researchers. This study examines the fundamentals of software cost and estimation. Different techniques of cost estimation should be employed when estimating costs. Estimates are made to find out the cost, to the developer, of producing a software system. Following are the literatures used for evaluation of the state-of-art work on the estimation of software Cost.

2.1 REVIEW ON ESTIMATION OF SOFTWARE COST

Increased use of information system has led to bigger amount of developing expenses which insists on software. Using SVR (Support Vector Regression), a sort of machine learning technique (Lee & Kwon 2009), the software cost is estimated approximately. This finds the best set of parameters applying immune algorithm. The Software cost estimation was performed by SVR based on immune algorithm while changing populations, memory cells, and the number of allele (two or more forms of a gene). Moreover, they have examined and compared the effects of software cost estimation by means of the optimal set of parameters found by immune algorithm and showed that software cost
estimation using SVR based on immune algorithm was superior to linear regression model and other machine learning methods like analogy, artificial neural network, and genetic programming.

Estimation of cost for any project was the most important and distinctive matter for the project managers because of lack of data during the early stages of software development and intrinsic uncertainties which are related with it. In spite of huge number of suggested algorithmic models, the science of cost estimation yet lacks accuracy because of their incapability to handle with these uncertainties. To triumph over this circumstance, concentration has been moved towards soft computing techniques. Kashyap & Misra (2013) have attempted to suggest a cost estimation model based on Multi-objective Particle Swarm Optimization (MPSO) to adjust the parameters of the familiar COstructive COst MOdel (COCOMO). They have also employed Quality Function Deployment (QFD) technique to promote a good understanding among customer requirements and design specification. This distinctive mixture will assist the project managers to competently plan the overall software development life cycle of the software product. Hence, it presents a better cost estimate. The combination of QFD technique and PSO methods are incorporated to increase better accuracy in software cost estimation.

Due to the difficulty of Service-Oriented Architecture (SOA), cost and effort estimation for SOA - based software development was harder than that of traditional software development. A framework has been suggested by Li & Keung (2010) based on Divide-and-Conquer (D&C) for cost estimation for building SOA-based software. Based on the law of D&C theory, that framework could be supportive for simplifying the difficulty of SOA cost estimation. In addition, both cost estimation modeling and software
sizing work could be satisfied correspondingly by switching the related metrics inside the framework.

For the competitiveness of software companies, the accuracy and reliability of the estimation of the software project effort was very significant. Good estimates perform an imperative role in the administration of software projects. Braga et al (2007) have brought in a method based on machine learning which provides the estimation of the effort jointly with a confidence interval for it. They have suggested the use of machine learning techniques combined with confidence intervals to enlighten the accuracy of estimates of software project effort. They account on a number of experiments by means of two datasets, which are planned to compare machine learning techniques for software effort estimation. The simulation effect shows that the suggested method was competent to build robust confidence intervals. This was of great significance for the user of a software effort estimation system, because such a system will offer an interval together with the estimation of the effort instead of the single estimation offered by traditional methods based on machine learning.

Magazinius & Feldt (2011) have verified intentional distortions of estimates reported in earlier study. This study was based on questionnaire responses from 48 software practitioners from eight dissimilar companies. The effects of the questionnaire propose that, presence of intentional distortions was affected by the organizational type and the software development process in use. The effects also propose that presence of distortions differs depending on the organizational factors, such as company type and development process (agile or plan-driven). These differences were further investigated by including more responses symmetrically divided between the comprised companies. Moreover, a future questionnaire should
comprise information on age, experience and role as it might cause the high standard variations in the effects of this study.

Lee et al (2010) have performed the calibration of the COCOMO II model by means of data gathered from the defense software development projects by employing Calico calibration software. The effect has pointed out that calibration could improve the precision of the original estimation model. They offer the effect of the empirical study on improving the precision of software cost estimation model for defense software development project applications. The results showed that estimation precision has improved after calibration. In addition they found that the performance of calibrating the multipliers and exponent as both was much better than calibrating multipliers only. The cause of the lower calibration performance and future works to improve the estimation precision is also discussed.

Software development effort is one of the most significant metrics that must be properly estimated. A hybrid method has been suggested by Bardsiri et al (2012) to raise the precision of development effort estimation based on the mixture of fuzzy clustering, Analogy-based estimation (ABE) and artificial neural networks (ANN) methods. In the suggested method, the result of unrelated and not consistent projects on estimates was reduced by planning a framework, in which all the projects were grouped. Two comparatively large datasets were used to assess the presentation of the suggested method and the attained results were compared to eight other estimation methods. These methods were chosen from the most common algorithmic and non-algorithmic methods employed widely in the field of software development effort estimation. Based on Mean Magnitude of Relative Error (MMRE) and prediction (PRED) (0.25) parameters,
comparisons were executed by means of three-fold cross validation
technique. According to the attained effects, the suggested method
outperformed the other methods and considerably enhanced the precision of
estimates in both datasets.

Tsunoda et al (2013) have spotlighted on an effort estimation
method based on early phase activities and its ratio to the effort of the
whole improvement process, and it assesses estimation precision of the
model by means of early phase effort. To attain the goal, they built effort
estimation models by means of early phase effort as a descriptive
variable, and compared the estimation accuracies of these models to the
effort estimation models based on software size. In their test, they
employed International Software benchmarking Standards Group (ISBSG)
dataset, which was gathered from software development companies, and
considered ‘planning phase effort’ and ‘requirement study effort’ as early
phase effort. The effect of the experiment demonstrated that when both
software size and sum of planning and requirement study phase effort
were employed as descriptive variables, the estimation precision was
most enhanced.

2.2 REVIEW ON VARIOUS ESTIMATION TECHNIQUES

All the estimation methods are studied and compared. Different
estimation techniques of current works are evaluated in this division.

2.2.1 Use case techniques

In the software development life cycle, Software effort
prediction was an imperative task. A lot of models including regression
models, algorithmic models, machine learning models, expert judgment and estimation by analogy have been extensively employed to estimate software effort and cost. A Treeboost model has been suggested by Nassif et al (2012) to predict software effort based on three independent variables which comprise software size in use case points, productivity and complexity. The Treeboost model was proposed through a series of 1,000 trees and was coached by means of 59 data points. A multiple linear regression model was formed and the Treeboost model was assessed against the multiple linear regression models with the use case point model by employing four performance criteria: MMRE, PRED, Median of Magnitude Relative Error (MDMRE) and Mean Squared Error (MSE). Experiment effect exposed that the Treeboost model could be employed with promising results to estimate software effort.

For the software industry, Software cost estimation is an open issue, which undergoes cost overruns regularly. As the most famous technique for object-oriented software cost estimation, Use Case Points (UCP) method, on the other hand, has two main disadvantages: the vagueness of the cost factors and the sudden classification. Fan et al (2009) have suggested the Extended Use Case Points (EUCP) method to address these two matters. The work constructs a probabilistic cost model EUCP, initiated from integrating fuzzy set theory and Bayesian belief networks (BBNs) with the UCP method, EUCP offer a probability distribution of cost and a refined gradual classification, which improve the vagueness of cost factors and develop the precision of classification. This provides two case studies to reveal the efficiency of EUCP in the real life.
Software estimations are based on prediction properties of system, with courtesy to development methodology. In object-oriented study, Use Case models explain the functional requirements of a software system, so they could be a basis for software measurement and sizing. Use Case points method that proposed by Karner (1993) was based on size and difficulty of Use Cases. Yavari et al (2011) has assessed the weak points of Use Case complexity metrics. They spotlighted on Use Cases and resources associated to Use Cases, and bring in other metrics for finding out Use Case complexity and next calculating unadjusted Use Case weight. These metrics could develop precision effort estimation.

In all aspects of project management, Global software project development needs new approaches. Azzeh (2013) has explored the applicability of Use Case Point estimation model to global software project development. This kind of projects was appropriate when local cost rate was high and the demand to release software to the market rapidly. This study has also demonstrated and emphasized the feasible significance role of UCP software cost estimation model in multi-site development. It examines the potential of Use Case Point estimation model for global projects and employs this as a basis to converse three suggested factors Global team trust, Global team composition and Culture value that will assist in managing the global software project development. Future extension of the suggested model was designed to reflect on the result of the different complexity factors.

2.2.2 Analogy based techniques

Software development cost overruns frequently make project managers to reduce manpower cost at the expense of software quality. Accurate effort estimation helps to avoid cost overruns. Analogy-based effort
estimation forecasts the effort of a project by employing the data of its related historical projects, where the resemblance is calculated using Euclidean distance. Wen et al (2009) have suggested that the accuracy and reliability of analogy-based software effort estimation model are developed by means of principal components analysis (PCA) and correlation weighting techniques. The experimental effects revealed that their approach considerably improves prediction accuracy and reliability over the traditional method, either by employing correlation weighting alone or by employing PCA combined with association weighting. The similarity of their approach with other approaches accounted in literature proposes that their approach was competitive.

Keung et al (2008) have suggested a means of software cost estimation as a substitute to other data-intensive methods such as linear regression. Disappointingly, there were disadvantages to the method. There was no mechanism to evaluate its suitability for a particular data set. Additionally, heuristic algorithms were essential to choose the best set of variables and recognize unusual project cases. They bring solution to these problems based on the use of the Mantel’s correlation randomization test called Analogy-X. They employ the strength of correlation among the distance matrix of project features and the distance matrix of famous effort values of the data set.

The software cost estimation affects nearly all activities of software project improvement such as: bidding, budgeting and planning, the accurate estimation was very important to the success of software project management. Li et al (2008) have suggested the Analogy Based Sampling (ABS) technique for predicting interval generation and compared it with the well-known Bootstrapped Analogy Based Estimation (BABE). The results and comparisons portray that ABS could increase the performance of BABE
with much higher efficiencies and more precise interval predictions. The point prediction accuracies were also compared in the study. The effects demonstrated that there were no important differences among ABS and BABE on point predictions. Additionally, the weighted features and its selections are inspected as they may have different relevance to the project development cost.

Kocaguneli et al (2012) recognized the necessary hypothesis of analogy-based effort estimation, the dynamic selection of the most nearest neighbor. They check that hypothesis by generating a binary tree of clusters of effort data and comparing the difference of super trees versus smaller subtrees.

For over a decade the achievement of estimating software project costs by means of analog-based reasoning has been obvious. Phannachitta et al (2013) have suggested ABE-CUDA, an algorithm for analogy-based software cost estimation. It employed the Compute Unified Device Architecture (CUDA) computing platform that speeds up the rigorous computational part of the cost estimation problem. For over a decade, analogy-based estimation (ABE) has been confirmed to be a helpful procedure for software cost estimation in spite of not being executed at its full predictive power because of the shortage of resources and lack of a supporting architecture to carry out the extensive searches required. Using 11 real-world datasets the method has been assessed from the PROMISE repository. Results have revealed that the suggested ABE-CUDA approach was competent to generate the best project cost estimates by finding out the best feature subsets and the most appropriate number of analogous projects for estimation, significantly improves the overall feature search time and prediction precision for software cost estimation. They designed to improve a
solution utilizing other estimation methods to generate results from multiple estimation models or more sophisticated and difficult techniques, which should be helpful to permit software developers to better know and estimate their project resource requirements.

To develop the similarity measurement (Phannachitta et al 2013) have empirically examined the use of probabilistic-based distance functions. The probabilistic-based distance functions were significantly more dynamic, as they gather the implicit correlation among the occurrences of project feature characteristics. This information gain, enables the constructed estimation model to be more short and understandable. They have investigated 6 distance functions based on probability estimation in an effort to develop software projects similarity measure, which directly manipulates the predictive power of software cost estimation by means of analogy. The authentication procedure revealed the accuracy improvement in all the 5 datasets with different functions employed. The results were further compared and confirmed by means of Wilcoxon signed-rank test for their statistical significance. In general, the effects were promising, and pointing out that probabilistic-based similarity measures offers considerable improvement over standard Euclidean similarity measures for applications such as Software Cost Estimation in Software Engineering.

Azzeh (2011) has suggested an adaptation approach by means of Model Tree based attribute distance to fine-tune Estimation By Analogy(EBA) and to get new estimates. Using Model Tree has a benefit to handle categorical attributes, minimize user interaction and develop competence of model learning through classification. Mohammad Azzeh has offered an improvement of the original formulation of the nonlinear adjustment mechanism that was more commonly valid to EBA. The
suggested method could now be used in the absence of feature selection and where features were negatively connected to effort. Experimental effects demonstrated that the suggested approach produced better results when compared with those attained by employing estimation by analogy based linear size adaptation, linear similarity adaptation, null adaptation and ‘regression towards the mean’.

A method which has been suggested by Keung & Kitchenham (2007) was based upon the use of the Mantel randomization test. Particularly, they find out project feature weights based on the strength of connection between the distance matrix of project features and the distance matrix of known effort values of the dataset. They have showed the procedure on a particular dataset, showing the use of the Mantel correlation to recognize whether analogy was suitable, and whether the project feature weights could be found out by statistical inference. Their results also reveal greater prediction accuracy during multiple project where employed with known weights. Therefore, they offer a sound statistical basis for analogy. It was one of the most important developments to analogy based software cost estimation.

Keung (2008) have brought in both the notion and the application of the Theoretical Maximum Prediction Accuracy (TMPA) for software cost estimation by means of analogy. They found out the TMPA of analogy by means of a distinctive dynamic K-NN approach to simulate and optimize the prediction system. They assessed their TMPA by means of the Desharnais and two random datasets. Their results have revealed that, using TMPA is an efficient approach to report the influence of each project’s impact to the overall estimate. The effects of an empirical experiment also reveals that their method was practical and significant for researchers.
seeking to increase enhanced prediction models, because it presents an alternative for practical comparison between different prediction models. Their method was hence a main development to the assessment of software cost Estimation models.

2.2.3 Work Break down based techniques

To sustain the prediction of efforts in the early stage of a software project development, Lee et al (2012) has suggested an estimation model along with a software effort estimation process, to estimate project efforts from high level WBS. The suggested estimation model was composed of four unique task assignment factors – team size, intensity, concurrency and fragmentation and one newly suggested factor, named ‘WBS effort’. The results point out that, the WBS effort factors could be engaged to develop the accuracy of effort estimations. This model has better predictive power for regression model. The accuracy of the prediction was enhanced by using the five mentioned factors. For the examined data, this model illustrates to have improved results in goodness of fit and quality of estimation.

Jing-jing et al (2013) has constructed WBS standard templates and dictionaries of Four Electrical (Communication, Electricity, Electrification, Signal) railway projects based on the methods of project cost management and Work Breakdown Structure (WBS) technique. They employ project software to set up four electrical standard resource databases for estimating the consequent cost and then, join the specific construction plans with standardization templates to explain the basic standard curve of project cost.
2.3 REVIEW ON SOFTWARE COST ESTIMATION METHOD

There are different methods to estimate the software cost. The below sections describe these different methods briefly.

2.3.1 Data mining based methods

In order to motivate confidence in a business setting, a predictive model was necessary to be precise and comprehensible. Dejaeger et al (2012) have tackled the problem by reporting on the effects of a large scale benchmarking study. Different kinds of techniques were under consideration, with methods inducing tree/rule-based models like M5 and Classification and Regression Trees (CART), linear models such as different types of linear regression, nonlinear models (Radial basis function networks, multilayered perceptron neural networks and least squares support vector machines) and estimation techniques that do not responsively induce a model (a case-based reasoning approach). The findings were subjected to careful statistical testing and it points out that, ordinary least squares regression in mixture with a logarithmic transformation executes the best. An added key finding was that, by choosing a subset of highly predictive attributes such as project size, development, and environment related attributes, naturally a considerable increase in estimation precision can be gained.

Software effort estimation needs high precision, but precise estimations are not easy to attain. There are a huge number of different methods combination exist for software effort estimation, choosing the most appropriate combination turns out to be the subject of research in the study proposed by Sehra et al (2014). Data preprocessing was executed and effort
was analyzed by means of COCOMO Model. After this process, data mining techniques - OLS Regression and K Means Clustering were executed on preprocessed data and effects attained were compared. Data mining techniques when executed on preprocessed data seems to be more precise than OLS Regression Technique.

2.3.2 Fuzzy based methods

The stage-effort estimation permits project manager to re-allocate exact number of resources, re-schedule project and control project development to complete on time well inside the budget. An approach has been suggested by Azzeh et al (2010) to use prior effort records to estimate stage effort. The suggested model combines ideas of Fuzzy set theory and association rule mining. The results were excellent in terms of prediction accuracy and have prospective to transport good stage-effort estimation. The purpose of the suggested method was to make sure whether the prior effort records could be employed to forecast stage effort with sensible accuracy. The attained results make sure that, employing association rule and Fuzzy set theory leads to considerable improvement in stage-effort estimation and provide project manager an evolving picture about project progress. Comparing this approach with exponential regression demonstrates that there was a significant potential in estimation precision.

A model has been suggested by Kumar et al (2011) by means of fuzzy logic in-order to estimate the most significant factors of software effort estimation such as, cost and time. From the empirical assessment they conclude that suggested fuzzy logic model demonstrates better software effort estimates considering the MARE assessment criteria as compared to the traditional estimation models. MATLAB has been used to find the
numerous parameters of software cost estimation. The performance of the model was assessed on published software projects data. Comparison of results from the model with existing universal models was revealed. The results showed that using the fuzzy logic method for the software effort estimation was a convenient approach to address the problem of vagueness and ambiguity survived in software effort drivers.

Hari et al (2010) have suggested a model named Interval Type 2 Fuzzy logic for software cost estimation. The inputs were fuzzified by employing Takagi-Sugeno fuzzy controller of Universe Discourse with mean and standard deviation of size values which affects the control performance. Rather than applying a single number, the software size could be considered as a fuzzy set giving the cost estimate in the form of a fuzzy set. The vagueness was an inherit part in cost estimation. They decreased the vagueness generated by the Type-1 functions by employing Type-2 Fuzzy logic. The model deals successfully with inaccurate and vague input and improves the dependability of software cost estimation. The estimated effort was optimized by means of the developed model and checked on NASA software projects on the basis of three criterions for evaluation of software cost estimation models.

A Successful project is the one that was completed and delivered on time, within the budget and with precise quality. The main issue in software project management are accurate software estimation such as cost estimation, quality estimation and the risk study. A common framework for software estimation has been offered by Saxena & Singh (2012). The framework contemplates on the Pre-processing Neuro-Fuzzy Inference System, the Neuro-Fuzzy Bank and Algorithmic models. Comparative Analysis study between Neuro-fuzzy model and the traditional
software model such as WalstonFelix, Halstead, Bailey-Basili and Doty models was presented. The assessment criteria were based upon MMRE (Mean Magnitude of Relative Error) and RMSE (Root mean Square Error). Integration of neural networks, fuzzy logic and algorithmic models into one scheme has resulted in offering robustness to inaccurate and vague inputs. Thus, the suggested Neuro-Fuzzy System was competent to offer good estimation capabilities.

Idri & Azeddine Zahi (2013) have authenticated and compared the Fuzzy Analogy and Classical analogy approaches for estimating the cost of Web hypermedia applications. Therefore, the suggested method plans to reproduce the effects of their precedent experiments on the dataset. In addition, questions concerning the estimate precision, the tolerance of indistinctness and vagueness of cost drivers, and the constructive context to employ estimation by analogy were discussed. This study granted the effectiveness of Fuzzy Analogy for software cost estimation.

Andreou & Papatheocharous (2008) have tackled the problem of software cost estimation through fuzzy decision trees, planning at attaining precise and dependable effort estimates for project resource allocation and control. The approach of Fuzzy Decision Trees (FDT) removes sets of association rules to utter relationships found among project attributes under exploration and effort. The experimental effects propose that the suggested approach may offer precise cost predictions in terms of effort. Furthermore, there was strong proof that the fuzzy transformation of cost drivers tend to improve the estimation process.
2.3.3 Genetic Algorithm based methods

For software project management, software cost estimation was important. Several approaches have been suggested to estimate the cost with present project by referring to the data gathered from previous projects. Li et al (2007) have brought in the methodology for appropriate projects (Project Selection) to improve the performance of ABE by adopting Genetic Algorithm. In order to attain successful results from ABE, several earlier studies suggested effective methods to optimize the weights of the features (Feature Weighting). On the other hand, ABE was still criticized for the low prediction precision and the sensitivity to the outliers. The assuring results of the suggested Project Selection for Analogy Based Estimation (PSABE) methodology was compared with other ABE methods and many frequently active machine learning techniques such as (ANN, Radial basis Function (RBF), and CART) points out the efficiency and possibility of the approach and its potential as successful method for Software Cost Estimation.

A hybrid intelligent method has been offered by Araujo et al (2010) to plan Morphological-Rank-Linear (MRL) perceptrons to work out the Software Development Cost Estimation (SDCE) problem. The suggested method employs a Modified Genetic Algorithm (MGA) to find out the suitable particular features to develop the MRL perceptron performance. The suggested method employs the MGA to find out the best particular features to develop the MRL perceptron performance, with the initial parameters of MRL perceptron. In addition, a gradient steepest descent method was employed to optimize the MRL perceptron parameters provided by MGA for each individual of MGA. An experimental study was performed with the suggested method by means of the Desharnais and COCOMO databases. In
the tests, two related performance metrics and a fitness function were employed to evaluate the performance of the suggested method.

Several methods have been suggested for cost estimation in past decades. Li et al (2008) have suggested Genetic Algorithm to concurrently Optimization of the K parameter and the feature weights for ABE (OKFWSABE). The suggested OKFWABE method was authenticated on three real-world software engineering data sets. The experiment results revealed that their methods could considerably develop the prediction precision of conventional ABE and have the potential to turn out to be a successful method for software cost estimation. Additionally, the experiments propose that OKFWABE’s performance may be helded back by high degree of non-normality.

Accurately forecasting software effort is vital for software developers. Underestimated software effort in the early phase might cause a severe effect. It not only influences the program, but also advances the cost price. It might cause a vast shortfall, because, all of the different software development team has it was own method to work out the software effort. In order to work out these problems, Lin et al (2011) have suggested a model which joins genetic algorithm (GA) with support vector machines (SVM). They could locate the best parameter of SVM regression by the suggested model, and makes more precise prediction. During the research, they check and authenticate their model by employing the historical data in COCOMO, Desharnais, Kemerer, and Albrecht. They demonstrate the effects by prediction level (PRED) and mean magnitude of relative error (MMRE).
2.4 REVIEW ON OTHER ESTIMATION METHOD

2.4.1 Neural Network based methods

In software project management, Software effort estimation was an important phase. COCOMO was one of the most employed models which have a parametric form. In addition, artificial neural networks (ANN) are joined with COCOMO and these methods raised overall performance. Sarc & Duru (2013) have suggested a method which combines COCOMO employed ANN with K-Means which is applied to estimate effort and feasible boundaries. ANN output was employed as input to K Mean sets and best set value was computed with possible lower and upper effort estimation values. This model is considerably better than COCOMO and ANN estimates. Experimental results reveals that suggested method has satisfactory results over ANN and COCOMO.

Attarzadeh & Ow (2010) have offered a model for handling unclearness and ambiguity by employing the neural networks. The purpose of the suggested work was to offer a technique for software cost estimation that executes better than other techniques on the precision of effort estimation. The suggested neural networks model demonstrates better software effort estimates considering the MMRE, Pred (0.25) evaluation criteria as compared to the traditional COCOMO. From the experimental results, it was concluded that, by the suggested neural network model, the precision of cost estimation could be enhanced and the estimated cost could be very near to the actual cost. In addition, the neural networks model offered better estimation precision as compared to the COCOMO model.
There are many cost estimation models. The most extensively employed model was Constructive Cost Model (COCOMO). Kaushik et al (2012) have suggested the employment of back propagation neural networks for software cost estimation. The model employed identity function at the input layer and sigmoidal function at the hidden and output layer. The model integrates COCOMO dataset and COCOMO NASA2 dataset to coach and to check the network. The test results from the trained neural network were contrasted with that of the COCOMO model. From the experimental results, the combination of the conventional COCOMO model and the neural network approach develops the cost estimation precision and the estimated cost could be very close to the real cost.

In finding out the success or failure of the product the estimation of effort engaged in developing a software product performs an imperative role. Project managers need a dependable approach for software effort estimation. Satapathy et al (2013) have suggested a model which is executed by means of Multi-Layer Perceptron (MLP) and Radial Basis Function Network (RBFN) and produced results has been compared. In addition, a relative study of software effort estimation by means of MLP and RBFN has been offered. The results reveal that MLP model provides less value of MMRE, NRMSE and higher values of prediction precision. Therefore it was concluded that the effort estimation by means of MLP model will offer more precise results than RBFN.

Patil et al (2014) have offered a new idea where Principal Component Analysis (PCA) works with Artificial Neural Network (ANN) by keeping the base of Constructive Cost Model II (COCOMO II) model. Feed forward ANN employs delta rule learning method to coach the network. Coaching of ANN was based on PCA and COCOMO II sample dataset.
repository. PCA was a kind of classification method, which could strain multiple input values into some definite values. It also assists in decreasing the gap between the actual and estimated effort. The suggested technology raises the correctness of the estimates without worsening the changeability, which develops the precision, turnaround time and presentation of the system.

Software effort assessment was an imperative aspect which comprises amount of cost, schedule, and manpower requirement. An Artificial Neural Network (ANN) prediction model that incorporates with Constructive Cost Model (COCOMO) has been suggested by Dan (2013) which was enhanced by using Particle Swarm Optimization (PSO), PSO-ANN-COCOMO II, to offer a method which could estimate the software development effort precisely. In addition, it keeps the benefits of COCOMO model, developing the precision of software effort estimation and declining the error rate. The results points out that, compared with original ANN-COCOMO II model, the precision of software effort estimation has raised 3.27% by using PSO-ANN-COCOMO II model.

Khosravi et al (2011) have suggested a quick, yet dependable method for the construction of Prediction Intervals (PI) for Neural network (NN) predictions. The suggested Lower Upper Bound Estimation (LUBE) method establishes an NN with two outputs for estimating the prediction interval bounds. NN coaching was attained through the minimization of a suggested PI-based objective function, which wraps both interval width and coverage possibility. As the suggested cost function was nonlinear, complex, and non-differentiable, a simulated annealing (SA) method was used for the minimization of the suggested prediction interval based cost function and training of the NN model. The expressed results for 10 benchmark regression
case studies obviously reveals that the LUBE method is competent of generating high-quality PIs in a short time. The quantitative comparison with three traditional techniques for prediction interval construction exposes that the LUBE method was simpler, faster, and more dependable.

2.4.2 Clustering based methods

In software projects, Software improvement effort was one of the most imperative metrics that must be properly estimated. Bardsiri et al. (2012) have suggested a hybrid method to reduce the effect of inconsistent projects, which leads to the raise of estimates precision. In the suggested method, the effect of unrelated and inconsistent projects on estimates was reduced by planning a framework, in which, all the projects were grouped. The quality of training in ANN and the constancy of historical data in ABE were enhanced by means of the suggested framework. The promising result demonstrates that the suggested method outperformed the other methods on both datasets. A structure including training and testing stages was planned to demonstrate that how ABE and ANN could employ the clusters and how these methods could be joined to erect the hybrid estimation method.

Software Cost Estimation could be explained as the process of predicting the most sensible effort necessary to finish a software project. Mittas & Angelis (2013) suggest a statistical structure based on a multiple comparisons algorithm which ranks numerous cost estimation models, recognizing those which have important differences in precision, and grouping them in non overlapping groups. The suggested framework was used in a large-scale setup of comparing 11 prediction models over six datasets. The results demonstrated the advantages and the considerable
information attained through the systematic comparison of substitute methods.

2.4.3 Other methods

Jorgensen & Shepperd (2007) planned to offer a basis for the development of software estimation research through a systematic review of earlier work. A Web-based library of these cost estimation researches was offered to ease the recognition of related estimation research results. The assessment results linked with other knowledge offer support for recommendations for future software cost estimation research.

To distinguish the necessary content of Software Effort Estimation (SEE) data, the least number of features and instances essential to detain the information inside SEE data. Kocaguneli et al (2013) have revealed that least SEE datasets employed in their research have small necessary content and the value included in using complex methods was restricted. The objective of the research was to explore the necessary content of SEE datasets and make recommendations considering which estimation methods (simple or complex) should be favored. Their effects have revealed that the necessary content of SEE datasets was amazingly small. Even the most commonly studied datasets could be summarized by a tiny portion of their characteristics and examples. It also points that such a decrease, guards the estimation performance.

The intention of Kitchenham et al (2009) was to find out under what circumstances individual organizations would be competent to depend on cross-company-based estimation models. They executed a systematic assessment of studies that compared predictions from cross company models
with predictions from inter-company models based on the study of project data. Experimental procedures employed by the studies varied, making it not possible to carry out formal meta-analysis of the results. It also concludes that further studies were required, however they must be independent (based on different data bases or at least different single company data sets) and should address particular hypotheses regarding the conditions that would support cross-company or within-company models.

Lokan & E. Mendes (2009) have examined the employ of two dissimilar types of chronological splitting when building effort prediction models for software projects. The plan of the study was to contrast the two types of chronological splits against each other, to observe whether either leads to improved prediction precision. Estimation models were constructed and assessed by means of training and testing sets formed using project-by-project chronological splitting and date-based splitting. Data from Release 10 of the ISBSG repository were employed to answer the research questions. They found no considerable differences among the precision of estimates produced with the two approaches. They have examined a huge and varied data set to know if the similar finding was seen with smaller and less varied data set or with a more informed basis for selecting splitting dates, was a theme for further research from this study.

Software development effort estimates are often too low, which may direct to poor project plans and project failures. The reason for this bias is that, the estimates produced by the developers are affected by the information which is not relevant for the use of effort. Jorgensen & Grimstad (2012) have demonstrated to obtain an improved considerate of the underlying mechanisms and the strength of this type of estimation bias. The most exciting finding may be that, the estimation bias rose considerably with
higher levels of interdependence, i.e., with stronger emphasis on connectedness, social context, and relationships. They have suggested that this connection may be facilitated by an activation of one’s self-construal when involving in effort estimation, and a link between a more interdependent self-construal and increase in search for indirect messages, lower ability to disregard unrelated context, and a stronger prominence on socially enviable responses.

A technique has been suggested by Azath & Wahidabanu (2011), where the software effort could be successfully estimated by employing Function Points (FP). The quality of the effort estimation process could be calculated by employing the ISO 9126 quality metrics along with COCOMO. The sole difference between the suggested and existing estimation of effort for the software system development was the level of quality deliberation, where, the effort could be estimated by using the minimum number of quality factors in existing methods, but the suggested effort estimation method covers the ISO9126 quality factors, which routinely replicates in the improvement of software. From the given results, the suggested method successfully estimates the effort of the software project models.