CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

One of the worst estimated attributes seen over the past few years is the software development effort. Only less than one quarter of the projects has been estimate that has come through without significant changes (Jorgensen & Shepperd 2007, Khatibi & Jawawi 2011). In order to complete the projects on time and within the budget, there is a need to reliably estimate effort (Jones 2007). Software cost estimation plays a significant role in software engineering practice predicting the success or failure of contract negotiation and project execution. The deliverables of cost estimation like effort, schedule, and the requirements constitute valuable information pieces for project formation and execution. They are employed as key inputs for project bidding, allocation of budget, project planning, progress monitoring, and all effort connected with the project.

In 2007, unreasonable and unreliable estimates are the major causes of project failure as seen by a Comp TIA survey of 1000 IT respondents. It finds two of the three most-cited IT-project failure causes and is concerned with unrealistic estimation of resource. The community of software engineering has put tremendous effort to recognize the significance of software estimation and also for generating reliable cost estimates for software estimation. Mean for prediction of the amount of effort and time required to complete the software. It helps determination of resources to be employed for completing the projects on schedule. It also helps in better
planning of software development (Neumann and Santillo 2006). Knowledge of some information of the background about the estimate before using it (Grimstad, et al. 2006) is necessary.

Several methodologies have been established for cost estimation; these are broadly classified into two following models:

1. Algorithmic models: these comprise linear models and power function models.

2. Non-algorithmic models: these include expert judgment (Leung and Fan 2002).

One of the base measures is the software size employed for software effort estimation. There are various aspects of the measurement of software size with each measurement having its own purpose. It can better be defined as a set of following attributes (Fenton and Bieman 2014):

- Length
- Functionality
- Complexity

Source Lines of Codes (SLOC) are used to measure the length of the program source code (Fenton and Bieman 2014), whereas the software functionality is often measured in terms of function points. There are several benefits of using function points over SLOC for estimating effort (Jorgensen and Shepperd 2007). Instead of using SLOC, it is better to employ functional size as an input for the purposes of estimation. For several years, measurement of software size remains a challenging area in the software engineering field.
Various problems are associated with measurements of software size and some of the key factors for reliable measurement of software size are (Buglione and Gencel 2008, Gencel and Demirors 2008):

- Employment of software size as an input for estimating the cost and effort.
- Analysis of planned size and actual size are analyzed for monitoring the achievements of project.
- Designing of contracts by using the measurements of software size and are depending on the unit amount cost of software size.
- Employment of software size in normalization of other measures.

The following five approaches are certified by International Organization for Standardization (ISO) as international standards for measuring the functional size of software.

- MK II Function Point Analysis (FPA) (Abran, et al. 2011)
- International Function Point Users Groups (IFPUG) FPA (Cuadrado-Gallego, et al. 2010)
In the field of software engineering, every time the effort estimation has been a controversial issue between researchers. It is highly necessary to make a relationship between functional size and effort for estimating the effort accurately. There are various other factors (Ahmed et al 2005, Chiu & Huang 2007, Huang & Chiu 2006, Jørgensen 2005, Li et al 2007, Park & Baek 2008) that influence the effort to be utilized apart from size and are as follows:

- Type of Application,
- Type of Architecture,
- Programming Language,
- Size of team,
- Quality of software and
- Platform.

Function points have some specific fundamental flaws in their construction seen in the utilization of valid measures (Kitchenham 1997). It is highly necessary to appropriately measure the function points for avoiding any unpredicted behaviour.

2.2 VARIOUS E-LEARNING APPROACHES

With the increasing usage of the internet, conventional class rooms are shifted to e-learning. The advancement of Information and
Communication Technology (ICT) has made e-learning methods more adaptable.

Computer based teaching has become the simplest and the easiest platform to communicate with the learners via virtual classroom, chat, email, interactive web based environment. E-learning refers to a learning environment where the teachers and students are separated by space and time or both. (ManopPhankokkruad 2012) The teacher is able to provide the digital course content through course management applications, multimedia resources, video conferencing, internet etc. Students can view the course content through internet at any time anywhere in the world.

In the e-learning environment training can be easily organized, scheduled more quickly through online than in the conventional classroom-based training. The sessions are easily scheduled or rescheduled and can be easily communicated, to the participant. The main advantage of the e-learning is that the learners need not travel to the venue of the session.

For accessing the e-learning contents like video, audio, animated pictures, chatting, e-mail and virtual classroom, the user needs more bandwidth of internet to access the course material. Cloud computing is the paradigm shift for the information technology and e-learning environment. A huge volume of resources and the content can be easily stored and retrieved through online via cloud computing on demand.

Currently many universities and educational institutions are moving towards either a private cloud or a public cloud. For deployment of the e-learning course ware in the cloud environment, facilities like videos, simulation etc., can be easily shared and viewed by the audience. The
demands are increased; and there is a need to introduce innovative techniques for the educational environment.

Various view-points emerged about the development, maintenance, and conception of e-Learning solutions. This approach (Dodero, et al. 2014) summarizes the following view-points:

- Automated methods,
- Integration of various approaches,
- Emphasis on human and social aspects,
- Domain-specific development methodologies,
- Model-driven or language-driven development and system integration techniques, and
- Grammar-oriented development.

All these points of view have a common goal for facilitating the growth of complex e-Learning applications and solutions. It can be accomplished by multi-disciplinary teams such as software developers, instructors, domain experts, students and final users.

At the university, this approach (McGill, et al. 2014) makes analytical study of the conditions associated with persistence of e-learning initiatives. From a survey of available literature, conditions associated with several issues are identified (McGill, et al. 2014). Initiatives reported between 2000 and 2008 in the e-Learning development are characterized among the initiatives of continued and non-continued learning. In both these conditions, the terms associated with learning and student response see agreeable presence. Alternatively, these initiatives were seen to provide much advantage
to the university in terms of finance. The circumstances that differentiate between continued and non-continued initiatives are dominated by technology characteristics and financial support. To support continuation, the technology required to be up to date and also best able.

In this method (Zaharias and Poylymenakou 2009), the growth of a questionnaire-based usability evaluation approach for e-learning applications is defined. The approach extends the existing practice by concentrating not only on cognitive but also e-learning usability. According to an established methodology in HCI research, it integrates web and the parameters of instructional design. Further, it associates them with an effective learning dimension that is intrinsic inspiration for learning.

In order to estimate the usability of e-learning courses offered in corporate environments, two large empirical surveys are conducted, resulting in valuable evidence for reliability and validity of the approach. Evaluation of the design of e-learning applications offers evidence for usability to practitioners.

E-learning has been implemented by corporations as the substitute training approach due to the cost, time, or flexibility for designer courses and learners. A huge effort has been made concerning about the evaluation of e-learning effectiveness. Nevertheless, a generalized quantitative evaluation model may consider the lacking of:

- Inter-affected relation between criteria and
- The fuzziness of subjective perception concurrently.

A Hybrid Multi Criteria Decision Making (MCDM) approach describes the independent relations and the dependent relations of evaluation
criteria with the help of factor analysis and of Decision Making Trail And Evaluation Laboratory (DEMATEL) respectively. Analytic Network Process (ANP) and the fuzzy integral approaches are employed for synthetic utility in terms of subjective perception environment. It results in effective e-learning evaluation programs with appropriate criteria. Thus, it fits with perception patterns of respondents (Tzeng, et al. 2007).

The e-learning environment, provides a number of online educational resources and communications with wide spread applications. Instructors are frequently overcome by the large number of messages generated by the students via online discussion forums. It is tedious and impossible for instructors to understand the progress of their students through reading and analysis of these messages. It results in adaptive teaching for a huge class, which is considered as a handicapped.

The key contribution of this approach (Lau, et al. 2009) is the design of a new concept map generation mechanism. It is reinforced by a fuzzy domain ontology extraction algorithm. Depending on the messages posted to online discussion forums, this mechanism can automatically build concept maps. It results in promising accuracy and better quality of the automatically produced concept maps. In order to improve e-Learning, this approach is employed for the development and application of the intelligent software tools. It also demonstrates the first application of fuzzy domain ontology extraction method for facilitating adaptive e-Learning.

This method (Cotton and Gresty 2006) discusses the think-aloud process in a current survey and also reflects on its contribution as a research methodology. A number of concerns about the approach arose during this study. They include the level of guidance given to participants, observer influence, and the data analysis complexity. But, the richness of the data
collected outweighs these constraints. The think-aloud approach has the potential to improvise the research in the software engineering field.

Several industries change their focus from capitalizing into processing power to hiring processing power from a specialized vendor as with rapid development of the architecture usage of cloud computing. E-learning systems usually necessitate various resources of hardware and software. There are several educational institutions that find such investments. Beyond their capability in because of the cost involved the cloud computing is the best solution for these type of difficulties. The implementation of e-learning system in cloud computing requires a particular method and also has its peculiarities. Upon e-learning solutions development, this approach measures the positive influence of using cloud computing architectures. A set of efficiency metrics in cloud computing is progressive for improved e-learning implementation. In the field of e-learning system, it evaluates the long term overall efficiency of the cloud computing usage (Pocatilu, et al. 2010).

2.3 PROBLEM WITH ESTIMATING EFFORT

- The relationship between effort and functional size is not yet fully understood. It can be reliable at distinct levels estimation and better project planning and management.

- Various project related factors like type of application, type of language, type of development, type of organization, and experience of the members were established to be among the most important ones.

- These surveys are made available on large datasets that include detailed information on various projects. International Software
Benchmarking Standards Group (ISBSG) dataset (Elyassami and Idri 2012) have been made available for the purposes of benchmarking and further research studies were conducted on the nature of the relationship between functional size and effort.

- Most of the projects are measured by the International Function Point Users Group (IFPUG) FPA technique as the Functional Size Measurement (FSM) approach. The nature of relationship between functional size and effort are analyzed (Jiang and Comstock 2007, Reddy, et al. 2010) for finding out the major factors that affect the relationship.

- It does not identify any single model after reviewing all the studies, which is recognized by the entire community. For the purposes of effort estimation, there is a need to construct a model and also validate the relationship between functional size and effort. These points are general issues for effort estimation.

Knowledge of the representation of software size is another relevant issue in the employment of estimation of effort. Conventionally, the total functional size is utilized in effort estimation models and can be represented in a number of ways and also as vector. Instead of using total functional size, Base Functional Component (BFC) types is investigated (Gencel and Buglione 2008) for effort estimation. These types can provide a more reliable estimate effort than the other techniques.

The Constructive Cost Model (COCOMO) is considered to be most common non-proprietary software estimation approach both in industry and literature. This approach was constructed using historical data and software development project assumptions. The features of this approach such as forms, cost factors and constants are applicable for software maintenance
cost estimation. The software maintenance is based on quality and complexity of the source code. This difference means reduced relevance and also low estimation accuracies. In order to evaluate the software maintenance cost, there is a lack of experimental studies, which calculate and prolong COCOMO or other models.

2.4 EFFORT ESTIMATION METHODS

Effort estimation is a method for calculating the time and manpower required for developing high quality software. This effort can be used to estimate the cost required for developing the software. It is the process of predicting the working hours and manpower needed to develop a project.

2.4.1 Types of Effort Estimation Methods

Different authors classify the effort estimation techniques from their own points of view. The commonly known are:

1. Empirical Parametric Estimation Models
2. Empirical Non-Parametric Estimation Models
3. Expert Estimation
4. Analogue Estimation Models
5. Downward Estimation
6. Upward Estimation.

2.4.1.1 Empirical parametric estimation models

These models are based on the knowledge obtained on the previous software projects to connect the size and effort value in terms of the explicit
function form. It is achieved by applying the regression analysis method, which is widely used. Benefits of this model are:

- Formalism
- Efficiency
- Objectivity and
- Fact

These factors are based on the knowledge drawn from engineering practice. Some of the negative factors are:

- Subjectivity of the input values
- Requirement for calibration before application in the concrete domain.

COCOMO model illustrates the hierarchy of three estimation models. It was suggested by Boehm et al (2004). This approach ranges from the macro calculation of estimating the product function value to the macro calculation with the structural examination. The most important motivation of the development of COCOMO model was to assist the realization of realize the cost of the decision consequence. It would assist to executing, formulating and supporting the software product. The three level model hierarchies are explained as follows:

**Model 1:** The *fundamental COCOMO model* is static single variable which is used to estimate the software development cost and effort as the function programmed size. It is expressed in estimated LOC.
**Model 2:** The intermediate **COCOMO model** calculates the effort based on the function of set of cost drivers and programme size which contains the subjective assessment of the following for attributes:

1. Hardware
2. Personnel
3. Product and
4. Project

**Model 3:** An **advanced COCOMO model** includes all the features of the intermediate model with the cost driver. They are all examined for each and every stage in the development process individually.

### 2.4.1.2 Empirical non-parametric estimation models

It is distinctive for empirical non parametric models which utilize the data on projects comprehended earlier. However, the estimation is not made still by applying the mathematical equations. Some of the models are neural networks, Optimized Set Reduction (OSR) technique and decision making truck.

Briand et al (1999)OSR pick up the subset of projects based on the productivity estimated for the new project. Here, productivity is defined as the effort based on man-month divided by the total number of LOC. The optimum subset grouped for the project should have identical cost factors as for a new project. Briand et al (1999) compared the accuracy measure for the COCOMO model and OSR approach to show the advantage the estimation models. Important OSR merits can be suitable to apply for the incomplete input data when the value of the subset cost factor is known.
Srinivasan and Fisher (2005) indicate two other non-parametric approaches for generating effort estimation. The first method is based on the self-learning algorithm to attain the decision making tree while the second method is based on neural networks. The experimental results show that the neural network results in smaller mean errors than the decision making tree model. But, the training of neural network is generally exhausting. The resultant accuracy of both the models is similar to the OSR model.

2.4.1.3 Expert estimation

Expert estimation models are based on the discussion of one or more person considered as experts in software development. Here, the Delphi technique is applied to coordinate the diverse opinions from the experts. Wideband Delphi approach support to discuss the problem with the experts. The implementation is based on the following procedure:

1. The software coordinator explains every aspect with the necessary project specifications
2. Coordinator arranges the meeting of experts to discuss the issues regarding the estimation.
3. Every expert independently and individually completes the form
4. Coordinator arranges another meeting to discuss the estimation that deviates most from the others
5. Software experts again complete the forms
6. Repeat step 4 and 5 until agreement has been reached with identify of views.
2.4.1.4 Analogue estimation models

Analogue estimation models need as much data as possible for project implementation. In some features, this model is systematized expert estimation model type and it is generally the experts who decide which project should the new project for compared. This model requires the estimation of the project characteristics that considered for similar projects. To concern a new project, effort, time and cost are used. The fundamental difficulties in this model are identification of projects on the basis of the similar estimation of the other projects. The best two analogue estimation models are:

- ESTOR and
- ANGEL

ESTOR (Mukhopadhyay, et al. 1992) is the kind of case based reasoning model and follows the five basic processes detailed below:

1. Objective case specification
2. Investigate for sufficient case to give out as original analogy
3. Relocate solution from the source case to target
4. Calculate the differences obtained between original and target case
5. Alter the initial solutions based on the dissimilarity found.

The components of function points and the inputs used on the intermediate COCOMO are the metrics used in the ESTOR. This model needs the basic analogy based on the function point value mechanism of projects. According to this ANGEL (Shepperd, et al. 1996) approach, projects are
discussed based on the function point components. The new project involves estimates based on the value of the neighboring projects. It can automatically estimate the best metrics subset on the basis of the input data. This model results in a mean relative 60% error whereas the regression linear has 226% error (Shepperd, et al. 1996).

2.4.1.5 Downward estimation

In this model, the total effort estimation is made based on the global characteristics of the software product (M.L. 1996). Usually, the downward estimation is based on the earlier software projects and considered into the account of all the project functions. The entire efforts are then distributed on the basis of the components.

2.4.1.6 Upward estimation

In the upward estimation model, the estimation is done based on the every project component. The entire effort is estimated in addition to the individual efforts (M.L. 1996). Approaches of this type result in many global effort components unnoticed such as system testing, project management and linked with integration.

2.4.2 Existing Effort Estimation Approaches

A challenge in reliable effort estimation is the uncertainty at the beginning stages of development process. An effort estimation method should take care of such uncertainty and so, Azzeh, et al. (2011) propose an analogy-based software effort estimation using fuzzy numbers. In this approach, an effort estimation method based on fuzzy numbers to improve the accuracy was proposed. This approach does not have the need to use all the numerical
attributes for selection. It obtains the fuzzy numbers from the attribute values instead of depending on unreliable expert opinions.

Rashid et al. (2012) explained about the application of case-based reasoning in the estimation of software development effort. A case-based reasoning model based on two similarity measures such as Manhattan distance and Euclidean distance is proposed in this research. Case-based reasoning model learns from experience due to its ease in preservation of a concrete problem instead of generalizing it. The actual development time was the input by the user with the parameters such as lines of codes, difficulty level and from the matching case, the predicted development time and error in prediction has been obtained.

Zia et al. (2012) define an effort estimation model for agile software development. User experience is used as a base for effort estimation. This model was developed to address various challenges to accommodate the agile method characteristics like adaption and iteration. The authors (2012) define an intelligent software effort estimation system. This method consists of user interface, language processor, knowledge and database. The user interface provides some predefined options like numeric values, whenever needed. The natural language processor translates the user response and query into specific rules. The inference engine was used to access the knowledge base of the input parameters which was given by the user.

Nassif et al. (2012) presented a method for effort estimation in the beginning stages of life cycle with the help of Cascade Correlation Neural Network (CCNN) model. It was a self-organizing network and contains an input, hidden and output layers. In this research, a novel correlation cascade model to estimate the effort from the use case diagram is proposed. The inputs are obtained from its size of the software, productivity and project
complexity. The size of the software is calculated using the Use Case Point (UCP) model and productivity is completed using the UCP model with some modifications. Every input is connected to the output and the neurons are added to the hidden layer one by one. The neurons are added to reduce the residual error.

Azam et al (2014) have determined a framework of software cost estimation by using the object orientated design approach. This method uses the available knowledge of the considered system as base for effort estimation. To represent the knowledge as base, this approach used an object oriented functional model. The effort is estimated by algorithmic and non-algorithmic approaches. This approach was used to obtain the parameters such as effort, software size and constraints. The effort includes person per month, number of hours and skills required. The software size includes Lines of Codes (LoC), function point and number of objects. The constraints such as budget, time, quality and hardware requirements are also obtained.

Popli & Chauhan (2013) define an agile software estimation technique based on regression testing efforts. It is considerable to consume time and cost in each and every iteration and so a regression testing based agile estimation approach was proposed. The total number of user experience anecdotes and points per user anecdote were calculated and from that baseline anecdotes point was obtained. Project related factors such as type, quality requirement etc. and people related factors such as communication skill, securities etc. were identified. The level set for each factor was determined and each factor is assigned with the value based on its level set. Then, the estimated story point is identified and velocity from its first iteration was obtained. Decelerated velocity was obtained from optimized velocity by considering various factors. From this, estimated time, cost and effort are obtained.
Aloka et al. (2011) tested the effort estimation using the Particle Swarm Optimization (PSO) based approach. This method was carried out to optimize test effort estimation thereby reducing the difference between actual and predicted effort. PSO was applied on two techniques such as use case point and test point analysis. This method involves steps such as initialization of swarm, application of PSO algorithm and analysis of the obtained output and only limited variants of PSO. Attarzadeh and Ow (2010) have depicted a soft computing approach for software cost estimation. This approach used a triangular membership function for cost and time estimation. This model involves three fuzzy steps such as inference from fuzzy rules, fuzzification and defuzzification process. It approach finds the efforts required for project based on the software size in kilo source line of code and also the other cost drivers called scale factors and effort multipliers.

Čeke & Milašinović (2015) surveyed early effort estimation in web application development. This approach aims to predict the effort of web application development before development but is not based upon the source code instead on the not yet developed application conceptual model. This model was created with the support of the simple linear regression equation. The dependence between web application models and functional size was also determined by this approach. Azzeh, et al. (2010) explained the fuzzy grey relational analysis for software effort estimation. In this approach, the impact of integration of the fuzzy set theory and within analogy based effort estimation was proposed. Some of the disadvantages of this method are: the distinguishing coefficient should not be the same for all datasets and should be varied to fit each dataset and each individual prediction. Another limitation resulting from using the fuzzy model was its requirement of a sufficient number of observations in order to construct fuzzy sets.
Reddy et al. (2010) have proposed a software effort estimation method using radial basis and generalized regression neural networks. A neural network was a massive parallel distributed processor made up of simple processing units. This network has a natural propensity for storing experimental knowledge and making it available for use. This approach determines Mean Absolute Relative Error (MARE), Variance Absolute Relative Error (VARE), Prediction, Balance Relative Error (BRE) and Mean Magnitude of Relative Error (MMRE). Shihab, et al. (2013) tested the appropriateness of the lines of code as a good measure of effort in effort-aware models. In this approach, LOC was empirically examined along with other codes and process metrics to predict which one gives the better results. Using a study of four open source projects, complexity measures were seen to have the highest correlation with effort. Further, that the combinations of LOC, code and complexity metrics were found to provides a better prediction than using the LOC alone.

Mittal et al (2010) have identified a software cost estimation method using fuzzy logic. The proposed model works on the following basis: Initially fuzzification and defuzzification was done. The measurement of software cost estimation model performance involves the measurement of Mean Absolute Relative Error (MARE), prediction and Absolute Relative Error (ARE). Choudhary (2010) have a definition for Genetic Algorithm (GA) Based Optimization method for software development effort estimation. This work investigated the inter-relationship among different dimensions of software projects such as models, project size and effort. In this method, the calculation of optimum effort was proposed. The procedural analysis of this model was carried out in two stages i.e. Conceptual View and Data Analysis.

A new approach in software cost estimation using a regression based classifier was defined by Gharehchopogha and Khalifehlou (2012). In
this research, an additive regression based approach for predicting the required cost and effort in software development cycle was proposed. This regression method was used for forecasting effort and time delivery of the project. The additive regression model determines the average value, which was the sum of separate terms for each predictor. This method was not applicable for parameters of non-algorithmic and algorithmic models. Gharehchopogh et al (2014) have proposed a particle swarm optimization approach for effort estimation. In this research, effort estimation was done by PSO algorithm and the process was as follows: Fitness was evaluated and new value for parameters were explored. The best values for the parameters were found from these and evaluated by the test dataset. Then, the estimated effort was evaluated and output was obtained on the satisfaction of the condition.

Zia et al. (2012) define a software cost estimation method using soft computing techniques. Software cost estimation was performed by a probabilistic approach which did not produce exact values. In this research has an effort estimation approach, which made use of fuzzy logic and PSO. Fuzzy sets were used for modelling uncertainty and imprecision and PSO was used for tuning parameters. Srivastava (2009) has presented an intelligent approach for estimation of software testing effort. A fuzzy based model for estimating approximate software testing effort was proposed. One of the greatest difficulties in using the model was determining and fine-tuning of fuzzy rules which depends on the exposure and experience of the decision maker. In this model, fixed triangular membership functions have been considered for the analysis and fuzzy rules have been derived. These fuzzy rules express the information for interpretation of the nature of software testing efforts. Identification of factors was done by a team of experts, which was flexible depending on the nature of the software requirement.
Nassif et al. (2013) have depicted an early software estimation model using a log linear regression and a multilayer preceptor model. In this research focused on software effort estimation from the use case diagrams using the use case point (UCP) model. In the UCP model, the unadjusted software size (UUCP) was calculated based on the complexity of the use cases. The adjusted use case point size (UCP) was then calculated by multiplying the UUCP by the technical and environmental factors. After the UCP size was calculated, software effort was estimated by multiplying the UCP size by 20. A multiple linear regression model was developed to predict the values of the productivity factor used in the proposed regression model. Additionally, a Mamdani fuzzy logic approach was used to adjust the values of the productivity factor. Another contribution in this research was the development of a Multi-Layer Perceptron (MLP) neural network model. This model takes the software size and the team productivity represented by eight factors as inputs. The output of this model is the software effort.

Rao et al. (2009) have proposed a novel neural network approach for software cost estimation using Functional Link Artificial Neural Network (FLANN). In this research discusses an approach for the validation of the dataset for training the neural network for the software cost estimation. The FLANN was used to predict software development effort (in person month) using popular algorithmic method called COCOMO. The FLANN network was used not only for functional approximation but also for decreasing the computational complexity. The use of FLANN to estimate software development effort requires the determination of its architecture parameters according to the characteristics of COCOMO.

Batra and Trivedi (2012) have proposed a fuzzy approach for software effort estimation. This method uses fuzzy sets rather than classical intervals in the COCOMO model for effort estimation. These fuzzy sets are
represented by Gaussian-shaped membership functions, which yield a reliable estimate. Capretz and Marza (2009) improved the effort estimation by using software estimation models. Considering the neuro fuzzy approach possesses lower MMRE, it was used in this approach. Adjusted function points and max team size were used in this approach along with development type, development platform, language type, and development technique.

Sheetz et al. (2009) have reviewed some methods for understanding developer and manager insights of function points and lines of code. This approach was to resolve whether a sensitivity gap subsists between managers and developers for function points and source lines of codes across several desirable properties of software effort measures. Analysis in this approach involves textual responses of four steps as follows: First two steps evaluated the responses independently and of keywords were identified that represents the concepts in the response of various participants. Second, two sets of keywords were combined to identify the textual responses.

Edagawa et al. (2011) have explained about function point measurement from Web application source code based on screen transitions and database accesses. In this approach, function point measurement was done by focusing on screen transitions in web application and database accesses. In this method, transition functions were extracted from screen transitions as a set of SQLs, and data functions are extracted from SQLs as a database table. It shows that the previous function points can be measured efficiently without a knowledge of the specifications.

Abrahão et al. (2010) has validated the size measure for effort estimation in model-driven web development. The Web development nature drives the project managers to spotlight on the time variable to achieve the required short cycle times. But this leads to several problems such as
exceeding budgets and bad product quality. Model-driven development approaches need to overcome this problem. In this approach, a measurement procedure for web applications, which is compliant with the function point analysis method, was proposed. The object-oriented hypermedia method provides the semantics and notation for developing web applications. The main feature of this method is its model-driven and transformation-based approach. The models are considered to be the most important artifacts in model-driven development. So, instead of predicting the volume of effort needed to develop a web application based on the source code, this method predicted it based on the corresponding conceptual models.

Efficient Software Cost Estimation using Neuro-Fuzzy Technique was defined by Khan, et al. (2009). In this research explains a general soft computing technique for software estimation. The technique has been checked against the algorithmic models like COCOMO and function point. PVGDP, et al. (2011) explain multi objective particle swarm optimization for software cost estimation. In this research new model proposes a new model for the estimation of the software cost. The multi objective particle swarm optimization methodology algorithm was applied for tuning the parameters. By using clustering method, the data items were divided into number of clusters and PSO could be used for parameter tuning of each cluster.

Seth et al. (2009) have defined a component selection efforts estimation method using the fuzzy logic based approach. There are five inputs to this fuzzy model, namely Reusability, Portability, Functionality, Security, and Performance. This model considers all five inputs and provides a crisp value of Selection efforts using the Rule Base. All inputs can be classified into fuzzy sets viz. Low, Medium and High. The output Selection Efforts are classified as Very High, High, Medium, Low, and Very Low.
Jeng et al. (2011) present a specific effort estimation method using function point. It is a different approach that trades “generic” with “specific” and greatly simplifies the estimation model to enable common programmers to use it. This approach possesses an advantage that, it makes the function classification more suitable for a particular application domain so that function point counting can be conducted by programmers themselves. Zheng et al. (2009) propose an estimation method for software projects effort based on function point. In this method, the function point was calculated as follows:

- The type of function point count was determined. Function point counts were associated with either projects or applications, which are of three types: 1. Development project 2. Enhancement project 3. Application.

- The counting scope and application boundary were identified. It was difficult to define where one application stops and another begins. So, the boundary lines were placed on a business view rather than on technical consideration.

- The data functions were counted to establish their contribution to the unadjusted functional point.

- The transactional functions were measured to find out their contribution to the unadjusted functional point.

- The value adjustment factor was determined and the adjusted function point count was calculated.
2.5 EFFORT ESTIMATION BASED ON COCOMO METHOD

The COCOMO is an open estimation model, in which all the details are published. It includes underlying cost estimation equations, with every assumption in the model, every definition and the costs incorporated in an estimate being plainly stated. The calculations are based on the estimates of a project’s size in Source Lines of Code (SLOC). The SLOC includes: the source lines that are delivered as the part of the product and the source lines obtained the project members.

Garg, et al. (2014) enhanced the COCOMO model using function point analysis to increase effort estimation capability. This method used regression formula for estimation by using historical data that has future and present characteristics. The COCOMO model proceeds from the design phase to the integration phase of cost and schedule of the project. In this approach, an intermediate COCOMO model is used in which extra feature of 15 cost drivers are added. Those drivers have fixed values that are multiplied to obtain an effort adjustment factor. COCOMO model is based on LoC and as the LoC increases, estimation becomes complex. So, this model was merged with function model to increase the accuracy.

Reddy and Raju (2009) have an improved the fuzzy approach for effort estimation by COCOMO model using Gaussian membership function. In this approach, fuzzy sets were used instead of the classical intervals in the COCOMO model and those sets were represented by Gaussian-shaped membership functions. Cost drivers of a software project were specified by distribution of their possible values, using fuzzy sets. Rather than characterizing the cost drivers using fixed numbers, interval values were used. These interval values were represented by various membership functions like triangular, trapezoidal etc.
Du et al. (2010) improved the software effort estimation using the neuro-fuzzy model with SEER-SEM. The neuro-fuzzy features of the model have the advantages of strong adaptability with the capability of learning, less sensitivity for imprecise and uncertain inputs, easy to be understood and implement, strong knowledge integration, and high transparency. The proposed neuro-fuzzy structure was used along with other algorithmic models besides the COCOMO model and presented effectively with various algorithmic models.

Attarzadeh & Ow (2010) propose a new software cost estimation model based on artificial neural networks. The effort drivers possess the properties of uncertainty and vagueness and so neural networks that can overcome those properties were used. This approach presented a method for handling imprecision and uncertainty by using neural networks on algorithmic and non-algorithmic estimation models. This method used a terminating condition and iteration was done until reaching that condition. This terminating condition is when all changes in weights are below some threshold or a specific number of iterations have been done.

Sharma & Verma (2010) have optimized a fuzzy logic based framework for effort estimation in software development. This approach provides a transparent, optimized framework for effort prediction, which was based on fuzzy logic. The Gaussian member functions handle the imprecision in inputs and make it a valid choice for representing the fuzzy set. Kazemifard, et al. (2011) propose fuzzy emotional COCOMO II software cost estimation (FECSCE) using multi-agent systems. Effort estimation was significant for effective project management. This approach uses social characteristics, emotion and personality factors, and the capabilities of the team member in order to have a better simulation of a real team. A multi-agent system was used to simulate intra-team communication. The difficulty
involved in the project was considered without change during project execution time.

Pytel et al. (2015) explain the about feasibility and effort estimation models for medium and small size information mining projects. There were two primary reasons for project failure such as unmanaged risks and inaccurate estimations in the needed resources. In order to handle these problems, two ad-hoc models have been proposed for use at the early stages of information mining projects. Attarzadeh and Ow (2009) present a software effort estimation approach, based on a new fuzzy logic model. This method depends upon two-sided Gaussian membership function in fuzzy technique later it was validated with collected data. This method has the advantage of good interpretability with the help of fuzzy rules. This research could be used along with the expert knowledge project data into one general framework.

Software effort estimation by use of case point and COCOMO model was defined by Nagar and Dixit (2012). In this method, to estimate the KLOC, the project was divided into modules and each module into sub modules till the KLOC could be established. Two methods for the efforts estimation was used i.e. one method used use case point and another one used COCOMO model. The use case explains about the functional requirement for the system and hence, the authors considered it as the best way to estimate the efforts.

Malik et al. (2013) have analyzed fuzzy approaches for COCOMO II. The input variables were changed to fuzzy variables using fuzzy sets for each linguistic value such as very low, low, nominal, high, very high and extra high depending on each cost driver and scale factor. A separate fuzzy inference system was designed for each cost driver. Rules were developed as
cost drivers in the predecessor part and the corresponding effort multiplier in the subsequent part. The defuzzified value for every effort multiplier was obtained from individual fuzzy inference systems.

Software effort estimation by COCOMO and function point analysis based on fuzzy logic approach were proposed by (Sheta & Aljahdali 2013). The problems in effort estimation were studied in this research. This was a challenging problem for a software project manager. The fuzzy logic was used as a soft computing technique and it was found to simplify the modelling process of the effort. This approach helped in creating two models based on fuzzy logic.

Sharma (2011) has analyzed a software cost estimation using COCOMO II. This approach show, how to make cost estimation using COCOMO II for a sample project, and outlines the basic steps, terms, and tools used. There were 17 effort multipliers that were utilized in the COCOMO II model to regulate the development effort. Balaji, et al. (2013) proposed software cost estimation technique using function point with non-algorithmic approach. Though the COCOMO model is an open one, it has some limitations. Thus, a fuzzy based estimation using the Triangular Membership Function has been proposed in this research.

2.6 SUMMARY

In this chapter, various techniques used for effort estimation of software development and e-learning development have been explained. This chapter also provides the various approaches used in effort estimation and an extensive survey of effort estimation approaches. The analysis is performed after the creation of design specifications and it takes time to complete the process. But, some of the approaches have disadvantages like depending on
the time spent on each phase and also ignores some hardware issues. To overcome the issues for developing the e-learning, the proposed effort estimation approach is focused.