CHAPTER 1

OVERVIEW OF SOFTWARE ENGINEERING

1.1 INTRODUCTION

Software Engineering is a discipline which is majorly concerned about development of systematic large software applications that are used in digital computers. This involves various theories and methodologies in designing which includes not only technical issues like tools and technologies for professional software development but also management aspects like scheduling, staffing, organizing, planning, controlling and budgeting. Software Engineering is a complex process which solves the problem by modeling it as an abstract model or also referred as iconic model. When compared with science SE is closely related to good practices in business. This well-perceived scientific method of software development takes user requirement as input and a software product is delivered as output.

This systematic and organized approach in software development was performed with a sequence of stages referred as Software Development Life Cycle. Some of the popular and commonly used SDLC models are agile development, waterfall model, V model, feature driven development model, several hybrid models, object oriented models, and various prototyping models. The structured set of tasks that are involved to develop the software system is called as a process. Each stage in SDLC is a process involving
resources, constraints and other tasks. They also impose consistency and help the programmers to improve the activities that are related with this.

The stages include gathering requirement from client, a detailed functional spec about what the system does and whether all the user requirements are fulfilled. When the customer accepts the functional specification as per the design specification the software is being developed and tested. The completed software is also delivered and installed in the acquirer end.

The major activities of a software process includes software requirement gathering, Software design, Software construction, Software testing, Software maintenance, Software configuration management, Software engineering management, Software engineering process, Software engineering tools and methods, Software Quality Assurance and Risk management.

1.2 BACKGROUND OF SOFTWARE ENGINEERING

Software Engineering is a systematic well quantifiable approach for design and development and maintenance of software. NATO conference of Software Engineering provoked the thought of SE at the time of crisis. Software requirement collection involves elicitation analysis, requirement specification and validation of software requirements. The software design process defines the involved architecture, components to be used, interfaces to be included and other sub components. Software construction is creation of code with high level programming language and necessary packages, these codes are verified, tested and debugging respectively. Dynamic verification process checks for good quality software with respect to correctness and
completeness. The intention of this is to check for errors in the developed software. The end result says it is defect free.

The ability to inherent change and control in a software project is Software Configuration Management. Change control process is continuous and requires frequent revision with certain constrains in mind. SCM guarantees the integrity, reliability and reusability of developing software products from beginning to release i.e throughout its life cycle. After the product is delivered support is extended by the industry for further performance improvement is done in the maintenance phase. The four classes of maintenance are adaptive, perfective, corrective and preventive. Proper computer based tools and methods are used for making the activity systematic leading to success of a project.

Among these difficult tasks, the challenging task given to a developer is not only in development but also in finding the development cost of software. Cost of software is discovered using the estimates termed as cost estimates or effort estimates. The relationship which exists between the software development cost and development cost charged to a customer is not that easy to predict especially in the beginning of the development. Pricing or development cost is primarily based on various factors they are Market opportunity is based on the brand name of the company mainly. If a company is trendy it may get more projects. In some case it may get similar projects or even new projects.

Requirement Volatility is a major avenue for increasing the price of software, the developer may charge with a claim about its requirement change. Financial health or financial status will also lower the profit margin on both sides if it is of sound health they can claim that meager profit is
sufficient. If it is severely affected it will even accept for small price gain to win a contract. Contractual term will also allow developer to reduce the price of the software, because the organization that developed will do the maintenance or will reuse code in other similar projects.

Software project Management focuses on a project from its commencement till its end with an eye on the following Cost, schedule and goals. Basically the life cycle of a project involves formulate and designing a project, planning, implementing and evaluating a project. Estimating the cost of a software product is one of the most difficult and error-prone assignment in software engineering. It is hard to find a precise cost estimate during the initial planning phase of software development with unknown factors. This makes to move toward higher degree of wishful thoughts towards estimation of effort.

1.3 CRITICAL PHASES AND ACTIVITIES

The critical phases of Software Engineering are analysis or study, design, implementation and testing. These are phases which guide in designing high quality software. The analysis or initial phase defines the customer requirements of the system when the definition was listed out without having a question in mind how it is being accomplished or in other words this definition does not include architectural or implementation information. A requirement document was prepared at the finishing stage of this phase. The customer can have a look at this document which provides exact and clear information. This requirement document was written in formal language specifies information with a degree of high level abstraction.

The design phase is also termed as second phase where in the architecture is customized and mapped in connection with requirement
document. Then a plan is put for further expansion into implementation which includes information about the OS, programming languages, machines, software packages, application architecture, distributed architecture layering, memory requirements, algorithms to be used, data structures, global type definitions, interfaces, and many other engineering standards are taken into consideration.

In the implementation, the team builds the software’s individual components and modules either from the scratch or tailored with a focus in mind about its quality, performance, testing and end deliverable. Lots of software-engineering projects testing are done separately by another team after implementation. One cannot look into his mistakes so if testing was done separately it was like redoing it obviously more errors are eliminated. Finally the end product is sent to a separate team of professionals for testing.

The important work in SDLC is effort estimation. Near about 60% of effort is spent on analysis phase and design phase. This shows the evidence of importance of effort estimation. The schedule could be prepared if human effort estimates are identified in the early development life cycle. Among these phases precise cost estimation is significant because:

- Overall business plan can be classified based on this.
- It helps to assess the impact of changes and helps in repeated planning.
- Customers expected cost and the actual will match in line.
1.4 PROCESS OF ESTIMATION

The projects to be built are based on the financial plan built up from manpower or labor, equipment and materials. Estimating Labor and equipment costs are assumed to be a challenging estimate. The cost of labor is dependent upon effort that has been put and cost of the labor per unit time. This effort put is also mainly dependent upon the activity size to be performed. The performed work is also mainly dependent upon the team member’s expertise. Cost estimation is primarily the product of work done and rate per hour.

Cost estimation is done by using the following steps.

- Set up cost estimation objectives
- Planning about the project resources.
- List out the software requirements and other requirements.
- Feasibility study on software system has to be worked out.
- Evaluate different estimates and iterate the software process.
- Once when the software project commences keep an eye on cost, progress and schedule.

Estimation model is chosen with the above constraints in mind. Calibration and assumption of the model, sensitivity of estimates and deviation of estimates are to be performed. This development of Software Engineering projects is too complicated process which includes financial risks and involves in-depth planning with a great amount of accountability in decision making. This is necessary to compute estimation of effort for project realization accurately and determine the project cost.
1.5 SOFTWARE EFFORT ESTIMATION

Developing software requires more of skill than any field of Engineering since it represents a complex and long lasting endeavor. The demand of software projects keeps on increasing continuously hence economic survival has entered into the scenario and it has become more important. Cost estimation of software generally includes amount of effort, duration, staffing level and other required inputs for its development. The prime challenge of software developers now is hence estimating the effort and schedule. These two are like sides of a coin and it is inter-linked. Software projects are vague and intangible in comparison with other projects. Hence the development cycle is complicated. Prediction of software effort is very difficult and could not be documented because of change in technology, developer’s knowledge and other attributes of project. Hence the development cycle is also too complicated. Over estimates and under estimates have a direct impact on the software organizations and it smashes the deal. Effort prediction is a function of several variables. The variables are the key contributors to effort. It is also very difficult to obtain such accurate information at the early development stage of a software project.

The potential benefits of accurately estimating development costs are large, especially when the vast amount of money spent on new and legacy software system is also considered; yet it is widely recognized that few companies are proficient at estimating effort. Basically in any statistical model, decisions are drawn from the study that reveals it is reliant on the historic input. When similar input samples occur the concluding results are drawn directly. If the inputs are different which are not available in the given sample, the conclusion is also not dependent on the given set of historic samples even if we get it could not be compromising. Contemporary
methodologies like empirical COCOMO are limited because of inability to manage change in internal and external surroundings in connection with software projects. This COCOMO is a parametric model which uses dependent variables includes schedule and cost based on quantitative indices. Often, parametric models can be distinguished and fine-tuned for explicit projects or projects inside specific software industries (Rad 2002).

Software effort estimation has been a major problem. It is difficult to reliably estimate the effort due to the following reasons.

- Lack of historical database for cost measurement
- Factors that affect effort and productivity are not understood well. Their relation is also need to analyze
- Unqualified and untrained staff for estimation
- Little penalty for the poor estimate

Major role associated with any software project’s success or failure of projects depends on effort estimation. Boehm and Papaccio’s literature about cost estimation of software concludes that “understanding and controlling software costs is extremely important”. In this research an extension of well known empirical model COCOMO II is developed with Graphical and Genetic approach. This COCOMO II lays emphasis on the effort estimation which is evolved from several extensions of Constructive cost model.

1.6 SOFTWARE SIZING

The apparent size of the software has a straightforward relationship with the effort required to build it. Software size is the critical factor which
affects the cost of the software directly. The popular metrics which influence the software cost are Lines of Code, Function point, Software science (Proposed, Halstead), Feature point and object point.

1.6.1 Lines of Code

The delivered source of program code is called as Line of Code. Lines of Code is dependent on programming language is the most widely used software metric. SLOC measure is a count of the number of machine instructions developed and was the first measure applied to software. It is very easy to count the instructions at the initial stage. Usually, this metric is estimated by dividing the project into several modules and sub-modules until the size can be estimated approximately. This measure includes two types of SLOC.

- Logical SLOC
- Physical SLOC

The SLOC metric is supported by various cost estimating tools and historical data. Thus, SLOC is considered as the simplest measure to come up with a number. In general we can say that SLOC, considered in a vacuum, is a poor way to measure the value that is delivered to the end user. 1000 lines of code is KDLOC, it can be used to estimate a complex project. Alternative way to compare without regard to direct volume is to measure the complexity of the software. This can be done with Function Point Analysis which can measure the complexity of the programs inputs and outputs. LOC can be obtained only after the software project has successfully completed. Estimating at the early stage is very hard to compute. PERT (The Program (or Project) Evaluation and Review Technique) is a typical method for analyzing
the tasks in a project. PERT can also be used for estimation which uses the code size. With this PERT chart decision making is facilitated a lot.

1.6.2 **Halstead’s Theory of Software Science**

This model proposed by Halstead comprises of code length or the size of the program termed as LOC and the amount of required storage space. This theory is based on the hypothesis that involves manipulation of operators and operands which are unique namely n1 and n2. Actual length of the program is ‘N’ which is the sum of n1 and n2. This hypothesis also says N_h the estimated value of length as

\[
N_h = n_1 \log_2 n_1 + n_2 \log_2 n_2.
\]  
(1.1)

1.6.3 **Function Points**

Albrecht introduced a new metrics which makes use of functionality of the software. The count of different type classes makes the function points. The different classes are User input, user output types, inquiry types, internal file types and external file types. The initial function points is either used depending on the complexity of the project or based directly on the count. The extension of function points are full function points (FFP) and feature points.

1.7 **PRESENT DAY SOLUTION FOR EFFORT ESTIMATION**

Accurate software effort estimation is a challenge in front of software industry. All present day effort estimation models are centered on the following metrics (1) Quality of the software (2) Quantity or size of code (3) Time taken for implementation (4) Productivity and (5) Cost linked with the development.
Quality of the software is represented as the quantification of compliance to user requirements. Thus, in order to be able to measure quality, the degree of conformance must previously be made quantifiable. Quantity refers to a size of a software project. Time is expressed as a logarithmic function based on the effort. Productivity is a constant, which is fixed for an organization at the time of a project. Costs are calculated using a trivial function, based on the product of quantity and quality, divided by the corresponding productivity.

Realistic goals can be set only if effort estimation are done perfectly. The effort estimation approaches can be classified as expert based estimation technique, Dynamics-based techniques, formal estimation techniques, learning oriented estimation techniques, regression based estimation techniques, composite Bayesian estimation techniques and combination based estimation.

**Expert estimation** includes quantification and this is primarily based upon the verdict process; an individual expert measures the performance and consistency. This includes a bottom up estimation and group estimation. Delphi and rule based techniques are examples for Expert estimation techniques.

**Dynamics-based estimation** techniques clearly acknowledge that effort factors change over the life cycle of the system development life cycle they are dynamic over time. The most popular dynamic techniques are based on dynamics approach are Madachy’s dynamic model. These are good for planning and controlling but calibration is difficult.

**Learning oriented estimation** techniques train the network using historic patterns and heuristics by extrapolation from specific lessons. The
algorithmic parameter values are adjusted to produce better results automatically. Examples for learning oriented estimation techniques are CASE base, fuzzy Genetic Algorithms and Neural Network. Genetic algorithms find the best suitable alternates with criteria usually called as fitness function. This figure of merit usually called fitness can be either maximized or minimized. Starting with the initial population it generates offspring’s which survive to the next generation of population. ANN is a model based on biological neural networks. The network is made up of neurons which are connected with each other which are rule based. The learning process helps to discover the operating point. Even with imprecise data Fuzzy logic can solve problems.

**Regression based estimation techniques** include Ordinary Least Square method and robust is an improved version of standard regression method.

**Formal estimation model** works mainly on the use of mathematical formula. Hence quantification can be termed as mechanical. This includes Size based estimation, analogy based estimation and parametric models. The prominent formal estimation techniques are SEER, Checkpoint, COCOMO and SLIM.

**Combination-based estimation** is a hybrid model based on judgmental and mechanical blend of estimates from several sources.

A familiar approach to the estimation of the software development effort is a single variable function of project size. The equation of effort in terms of its size is calculated as follows:

\[ \text{Effort} = a \times (\text{Size})^b \]  

(1.2)
where ‘a’ and ‘b’ are constants. The constants are typically determined by regression analysis and applied to historical data.

1.8 CHALLENGES OF THE PRESENT DAY EFFORT ESTIMATION MODEL

It is simple to estimate the overall value of the effort multipliers after assigning the appropriate values as per the requirements. In the case of a software product the various metrics used to measure its size are LOC, function points or object points. But the complicated issue is to estimate impact of the effort multipliers, which plays a major role in the estimation scheme and causes for overestimation or underestimation of the software development effort. Human experts with historical data and based on their experience they had in the past they compute the effort. Machine learning (ML) approaches include Artificial Neural networks, Fuzzy Logic, analogous method, Bayesian belief networks and rule of induction. The underlying parameter in ML involved like number of neurons, number of hidden layers and others affect the ANN performance.

The difficulties in estimation mainly are

- Estimation is difficult because it is done hurriedly that too in the initial stages.
- At the start of any software project it is difficult to devise complete specification. This reliable specification and other required specification makes the intricacy.
- Any estimation process requires complete set of data for assessment. Even estimators would have not come across such situations may be in their life time. In certain cases handling large projects is a frenzied task.
Estimators estimate may fail in certain cases where consideration may not include the junior developers, less experienced etc.

Cost drivers are hard to conclude without finding the influencing characteristics it could be very hard to predict. In other words measuring the impact of cost drivers and effort multipliers.

Evolution in Information Technology and the methodology of development are problematic.

Finding the relationship between them misassumption complicates the estimation process.

Characteristics of the software to be developed and the development process makes the estimation complex in terms of abstraction level, innovative aspects etc.

SCE has to be looked from perspectives like sociological and psychologically which induces commitment to work in a group, leadership styles and so on.

In this view, this work is aimed at refining the cost drivers and scale factors handling mechanisms in COCOMO II estimation scheme. Missing or vague values of these estimates this solved using imputation. Imputation is the replacement of missing values in data with estimation techniques and assumptions. The major problem that hampers useful application of imputation methods is bias, when an estimator’s long-run average (expectation) differs from the quantity being estimated. The deviation becomes a danger. When this difference is systematical, the results of analyses may be biased and false conclusions are easily drawn.” (Huisman 2000) Definitely, an imputation method will be plausible and consistent,
reduce bias while preserving the relationship between items within the data, and can be evaluated for bias and precision (Sande 1982). Missing data brings frustration; imputation is one of the key strategies that researchers employ to fill in missing data in a given dataset. Solving such issues is very intricate to put into operation and it is problem specific. Imputed data is used in order to find a suitable replacement for better accurate analyses. Imputation is a method of modifying for lost information. Missing reactions to information articles is a normal situation in any survey. This missingness regularly happens since the respondent denies or is unable to furnish information for a particular field or many and also because of typographical mistakes. Missing information may likewise effect from mis-keying or by editing process.

1.9 MOTIVATION OF RESEARCH

The purpose of this research is to develop a protocol for determining how to handle missing data, especially in Software Engineering data sets. This extensive survey carried out in this research proved that the inherent shortfalls of the existing effort estimation models results in deviation in schedules and budgets. Estimation of effort uncertainty has established diminutive notice in the literature (Sheperd 2007). Pendharkar disparages traditional effort estimation practices “for not providing appropriate causal relationships for predicting software development effort” (Pendharkar 2005), Fenton squabbles that conventional models do not “provide support for risk assessment and reduction, inability to combine empirical evidence and expert judgment, inability to handle incomplete information” (Fenton 2000) and he also proposed Bayesian network approach to encapsulate influencing factors and depending variables.
Software size is one of the base measures which are used for software effort estimation. There are different aspects of software size measurement. Each measure has its own purpose. Software size can better be described as a set of following attributes (N. E. Fenton and S. L. Pfleeger 1997)

- Length
- Functionality
- Complexity

The length of the source code of the program is measured in Source Lines of Codes (SLOC). Functionality of the software is often measured as Function Points. There are many advantages of using Function Points and SLOC to estimate effort. It is better to use SLOC as an input for the estimation purposes instead of using FP. Software size measurement remains a very hot and challenging area in the field of software engineering.

Many problems are associated with software size measurements. Some of the main reasons for reliably measuring software size are:

1. Software size is a key measure that is used as an input in estimating cost and effort.
2. Planned size and actual size are compared to monitor the project achievements.
3. Contracts are also formed by using software size measurements. They are based on the cost of unit amount of software size.
4. Software size is also used in normalization of other measures.
Inaccuracy in Effort estimates may lead to project management failures in IT. These issues motivated to conduct research with significant improvements. The proposed work not only provides a rational improvement but also it provides an advanced improvement in terms of missing variables and scope extension. This resulted in an enhanced estimation schema which can support the managers in strategic level to perform cost estimation in an effective manner.

The stimulus of this proposed research consists two folds:

The potential benefits of accurately estimating development costs are great, especially when the vast amount of money spent on new and legacy software system is considered;

- On the other hand, yet the inaccurate estimation models are used by almost all the organizations by compromising in terms of economic and quality aspects;
- This significant issue motivated the research to formulate a new estimation technique with a better outcome.
- Actually we start with a brittle of COCOMO model which mainly depends on interpretation of a set of linguistic variables alone. Many influencing parameters need to be taken account of and these are uncertain. Uncertain parameters are modeled using Graphical and Genetic Algorithmic method. This uncertain parameter makes larger vagueness in the COCOMO.

From the above perspectives, the goal of this research is defined as to provide the estimating community with an enhanced approach to the
estimation practices, which might complement the present practices in a better way. This produces high-impact, high-quality research consequences on software effort estimation.

1.10 MODELS ADDRESSED

The dramatic numerous increases in development of software have resulted in focus on estimation. The swift increase is because of software environments and continuous change in requirement has resulted in predicting precise effort estimation in software engineering. Estimation methods are basically of two categories namely algorithmic and non-algorithmic models. Non algorithmic models are similar to that of human behavior which learns like us. Algorithmic models are purely based on mathematical equations, which works better if reliable data is present. This is a continuous activity which forecasts the amount of effort required to develop a software system (Boehm 1981). Later on this model had a major extension referred as COCOMO II (Boehm 2000b). Statistic techniques like step wise regression certainly has limitations which can address to small part of the possible solution space and leaving the complex relationship (Kok et al. 1990). Effort estimation models which are algorithmic has limitations when imprecise information is to be accounted and strong collinearity among parameters.

On the other side Fuzzy system has complex rules increases the degree of meaningfulness (Gray et al 1997). Hodgkinson and Garratt have set up the Neuro-fuzzy model for cost estimation as one of the vital methodologies for developing non-algorithmic models (1999). Estimation models based on Expert judgment takes advice from expert having extensive experiences on similar software projects. Since many parametric models faced with uncertainties but with little human effort it could be computed. Expert
judgment models face with a limitation on finding out the data and collecting the requirements. The sizing submodel does simple estimation on its size namely SLOC, FP and POP’s. Chidamber OO system has further given extension to POP model (Minkiewicz 1998).

The COCOMO II model was simpler but widespread with six major classes’ namely parametric model, learning based, regression based, expert judgment, dynamic based models and composite approach (Boehm et al 2000). COCOCMO uses Line s Of Code and Function Point as the two major metrics. If the above two metrics are available then mathematically and by using some experimental equations effort could be estimated promptly.

ANN models have ability to learn from examples but it fails certainly due to non tolerance of incompleteness and requires effort in training and setting up. Project failure is because of inclusion of unreasonable and unreliable estimates this was noticed by the review made by CompTIA the IT industry association (2007). The Standish research group mentioned in CHAOS report reveals the profound crisis connected with the future of the software projects (Jorgensen et al 2006). This also indicates that the cost overrun associated with it is 189%. In the last decade several studies show that the software project failure is because of improper planning, sudden decisions taken, inaccurate estimations, insufficient requirements (Galorath et al 2006). Even the other research work indicates that the root cause for project failure is especially due to inaccurate estimation (Jones et al 2007, Molokken et al 2005).

Though formal estimation models have a success story because of mechanical quantization process such as formula computation it was understandably difficult to believe on this method. Establishing of
relationship and specific knowledge of background is highly desirable in such models. Hence this work focuses on an efficient and reliable model for accurate effort estimation which eliminates the hindrances that are caused to the software development community.

1.11 CONTRIBUTIONS OF THE RESEARCH

The purpose of this research is to provide insight into how sophisticated imputation techniques are and this facilitates the understanding. It also integrates between statisticians and software engineers to make success on effort estimation. All decision making processes require quality data which is of greater importance for any validation process. This estimation process makes the participants to keep their promises which make them to survive. The organization to some extent can stay away from abandoned or cancellation of projects because of financial plan and schedule overruns. This research essentially provides the estimating community with an enhanced hybrid approach to the estimation crisis, which might complement present practices.

This research proposes a hybrid model which combines Graphic and Genetic approach and was investigated successfully. It has been found that the hybrid estimation model enhances the Performance of effort estimation in early stages of the software development life cycle. For several decades research has been improving with respect to several factors in development with trustworthy developments. This made to move two a combinational approach where in the brunt of it was identified, the effort multipliers are group into categories and improved if some parameters are obscure and hazy using Genetic Approach.
The categories are visualized as three edged triangle in the graphical form with classification namely optimistic and pessimistic group. This graphical model is constructed with number of non Nominal effort multipliers in the PG and the number of non Nominal effort multipliers in the OG. GA’s is a population based approach which uses alike operators specifically initial population is generated randomly, crossover for escalating diversity in the population, fitness selection, variation operators, mutation and selection process selects new individual. In this crossover is counted to be the foreground operation and the background operation is mutation.

A customized version of the well-known COCOMO model is provided to investigate the effect of the software development adopted methodology in effort computation. Results are proved to be helpful with the support of incorporating the basic knowledge of Graphical method along with the Genetic Algorithms. This makes the prediction more accurate in finding the development costs which is found to be the asset which is most valued by the top level management. The developed combinational effort models are able to offer superior estimation capabilities. The improvement has been proved in terms of the performance validation factors such as Magnitude of Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), Root Mean Square (RMS) and Relative Root Mean Square (RMS & RRMS).

The work presented in this Thesis also investigates the existing relationship between software size and development effort. This approach it uses input data from graphical method and optimizes it further using Genetic Algorithm. The main conclusion of this research is a viable hybrid estimation technique that would seem to offer some advantages over approaches including, improved accuracy, easier use of categorical features and an ability to operate even where no statistical relationships can be found.
1.12 THESIS ORGANIZATION

This thesis is organized as follows:

In Chapter 1 the overview of the problem domain along with the present day solutions and their challenges in the software effort estimation domain are enlightened.

Chapter 2 reviews the existing effort estimation approaches with their pros and cons. It summarizes software effort estimation algorithms and its performance. Besides summarizing results from estimation surveys, the main purpose of this chapter is to discuss methodical aspects.

Chapter 3 illustrates about the motivation, goals, objectives and the experimentation methodology of the proposed research.

Chapter 4 elaborates the Graphical Model and its formulation. This also includes refining of Effort multipliers, Cost drivers and Scale factors.

Chapter 5 describes the Genetic algorithm as the imputation model used for getting the optimized version of the effort estimation model. The hybrid model for estimation gives a wide explanation of domain features that might materialize in this perspective.

Chapter 6 explains the validation over the proposed effort estimation model along with the experimentation strategy and standard assessment models; in this direction several issues are addressed. The drawn results clearly prioritize that the requirement has been achieved.
Chapter 7 provides a review of the achieved results by pointing out the assumptions that were made and the limitations of various estimation practices in software development process and it also highlights the accomplished goals. It concludes the proposed research with a list of open issues which remains uncovered in this and the possible future directions.

1.13 SUMMARY

Finding accurate software estimation has been a bigger research challenge for quite a long period of time. Effort estimation is overwhelmed with as many problems especially inaccurate and uncertain effort estimates which are dominant and often optimistic. However any estimation model if counted to be successful in the software industry it has to be trusted by the practitioners if it has a sound credibility with reference to prediction. In all the estimation models there exists a high degree of influence on irrelevant attributes and also the software estimation attributes are wildly guesses based on human knowledge which is not precise nor clear and distinct. Especially in medium and large scale software projects all resources required for development are to be planned ahead off. The resources required are human, infrastructural and financial resources. The literature lies in large space and has been explored fully in chapter 2.