The accurate prediction of software development costs is a critical issue to make the good management decisions and accurately determining how much effort and time a project required for project managers as well as system analysts and developers. Estimation is defined as “The action appraising assessing or valuing” or “The process of forming an approximate notion of numbers, quantities, magnitudes etc. without actual enumeration or measurement”. From these definitions it follows that task of estimation is not easy to do precisely.

There are many software cost estimation methods are available. No one method is necessarily better or worse than the other, in fact, their strengths and weaknesses are often complimentary to each other. Estimating a effort required for software development is the most challenging and annoying job that requires expertise, experience as well as good understanding of process, project management, metrics and most important use of proper estimation models, tools and techniques.

Good software estimation models can significantly help the software project manager, project stakeholders to make informed decisions about bidding values, planning the project, resource management, and delivering the project on time within budget. However, if managers use inaccurate estimation model for making decisions that may be a recipe for disaster.

1.1 SOFTWARE ESTIMATION MODALITIES
The Software Engineering Body of Knowledge (SWEBOK) has identified Knowledge Areas (KAs) such as software requirements, software design, software construction, software testing, software maintenance, software configuration management, software
Software Engineering Management KA addresses management and measurement including Software Project Planning, which further addresses Effort, Schedule and Cost Estimation. Based on the breakdown of tasks, inputs and outputs the expected effort range required for each task is determined using calibrated estimation model based on historical size-effort data available, otherwise method like expert judgment is applied.

**Figure 1.1 Software Estimation Techniques**
The Figure 1.1 depicts estimation techniques, methods, tools and their categorization [15]. The functional form of software estimation models are determined by theory or functional form.

SLIM is based on Putnam’s analysis of the lifecycle terms of so-called Rayleigh distribution of project personnel versus time. In SLIM, Productivity is used to link the basic Rayleigh manpower distribution model to software characteristics of size and technology factors.

Checkpoint is knowledge-based software project estimation tool. It has proprietary database of about 8000 software projects. It uses Function Points as its primary input and focuses on areas that need to be managed to improve software quality and productivity. Checkpoint predicts effort at four levels of granularity: project, phase, activity and task. Estimates also include resources, deliverables, defects, costs and schedules.

The PRICE-S model was originally developed for use internally on software projects that were part of Apollo moon program. It consists of following three sub-models

- Acquisition Sub-model: forecasts cost and schedule
- Sizing Sub-model: facilitates estimating size
- Life-cycle Sub-model: predicts cost of maintenance and support phase

Foresight 2.0 is latest version by PRICE Systems for estimating time, effort and cost for commercial and non-military government software projects.

ESTIMACS stresses approaching the estimating task in business terms. It also stresses the need to be able to do sensitivity and trade analysis early on in terms of staffing/cost estimates and associated risks.

SEER-SEM has been evolved into sophisticated tool supporting top-down and bottom-up methodology. Its modeling equations are proprietary but they take parametric approach to
estimation. The Scope of this model is wide. It covers all phases of project life-cycle, from early specification through design, development, delivery and maintenance. It handles a variety of environmental and application configurations, such as client-server, stand-alone, distributed, graphics, etc. It also supports development modes such as object oriented, reuse, COTS, spiral, waterfall, prototype and incremental. It allows staff capability, required design and process standards and levels of acceptable development risk to be input as constraints.

SELECT Estimator was released in 1998. It is designed for large scale distributed systems development. It is object oriented, basing its estimates on business objects and components. It assumes incremental development life-cycle. The nature of its inputs allows the model to be used at any stage of the software development life-cycle, most significantly even at the feasibility stage when little detailed project information is known. In later stages, as more information is available, its estimates become correspondingly more reliable. The actual estimation technique is based upon ObjectMetrix developed by Object Factory. It works by measuring size of a project by counting and classifying the software elements within a project. Applying the qualifier and technology adjustments to the base metric effort for each project element produces an overall estimate of effort in person-days, by activity. Using the total one man effort estimate, schedule is determined as a function of number of developers input as an independent variable.

COCOMO cost and schedule estimation model was originally published in 1981. It became one of the most popular parametric cost estimation model of the 1980s. It has experienced difficulties in estimating the costs of software developed by following new life-cycle processes and capabilities. The COCOMO II research started in 1994 at USC to address the issues on non-sequential and rapid development process models, reengineering, reuse driven approaches and object oriented approaches.

Delphi technique was developed at the Rand Corporation in the late 1940s originally as a way of making predictions about future events. More recently, the technique has been used as a
means of guiding group of informed individuals to a consensus of opinion on some issue. Participants are asked to make some assessment regarding an issue, individually in a preliminary round, without consulting the other participants in the exercise. First round results are then collected, tabulated and returned to each participants for a second round, during which participants are again asked to make an assessment regarding the same issue. Abts and Boehm used the technique to estimate initial parameter values for Effort Adjustment Factors appearing in the glue code effort estimation components of the COCOTS integration model. This technique has been used by Chulani and Boehm to estimate software defect introduction and removal rates during various phases of the software development lifecycle. These factors appear in COQUALMO (COnstructive QUALity MOdel), which predicts the residual defect density in terms of number of defects/unit of size.

Learning-oriented techniques include oldest as well as newest techniques applied to estimation activities. Former are represented by case studies, among the most traditional of manual techniques, later are represented by neural networks, which attempt to automate improvements in the estimation process by building models that “learn” from previous experience. Case studies represent inductive learning, whereby estimators and planners try to learn useful general lessons and estimation heuristics by extrapolation from specific examples. They examine in detail elaborate studies describing the environmental conditions and constraints that obtained during the development of previous projects, the technical and managerial decisions that were made and final successes or failures that resulted.

Neural networks is the most common software estimation model-building technique used as an alternative to mean least squares regression. These are estimation models that can be trained using historical data to produce ever better results by automatically adjusting their algorithmic parameters values to reduce the delta between known actual and model predictions.
Dynamics-based techniques explicitly acknowledge that software project effort or cost factors change over the duration of the system development. Factors like deadlines, staffing levels, design requirements, training needs, budget etc. all fluctuate over the course of development and cause corresponding fluctuations in the productivity of project personnel.

Regression-based techniques are used in conjunction with model-based techniques and include standard regression, robust regression etc. Standard regression refers to the classical statistical approach of general linear regression modeling using least squares. This regression technique is used to calibrate COCOMO II.1997. Robust regression, alleviates the common problem of outliers in observed software engineering data. Least Median Squares method fall in this category. Parametric cost models such as COCOMO II, SLIM, Checkpoint etc. use some form of regression based techniques due to their simplicity and wide acceptance.

Composite techniques incorporate a combination of two or more techniques to formulate the most appropriate functional form for estimation. Bayesian analysis is mode of inductive reasoning that provides a formal process by which a-priori expert judgment can be combined with sampling data to produce a robust a-posteriori model information in a logically consistent manner in making inferences. This is been used in COCOMO II.

1.2 MOTIVATION

Software researchers and practitioners have been addressing the problems of effort estimation for software development projects since at least the five decades. Most of the research has focused on the construction of formal software effort estimation models. The early models were typically based on regression analysis or mathematically derived from theories from other domains. Since then a high number of model building approaches have been evaluated, such as approaches founded on case-based reasoning, classification and regression trees, simulation, neural networks, Bayesian statistics, lexical analysis of requirement specifications, genetic programming, linear programming, economic production models, soft computing, fuzzy logic, statistical bootstrapping, and combinations of two or more of these
models. It is evident from studies that human expert can estimate with limited accuracy if textual information available during initial stages of project.

The accuracy of a technique is a function partly of the technique, partly of whether the technique is being applied to a suitable estimation problem, and partly of when in the project the technique is applied. Some estimation techniques produce high accuracy but at high cost. Others produce lower accuracy, but at lower cost. Normally one will want to use the most accurate techniques available, but depending on the stage of the project and how much accuracy is possible at that point in the Cone of Uncertainty, a low-cost, low-accuracy approach can be appropriate.

It is reported that in spite of enormous attempts, researchers and practitioners couldn’t make consensus over versatile methodology for software estimation particularly at initial stages of software project life cycle. The important causes of this problem are

- inherent uncertainties with project,
- incomplete and vague requirements,
- personnel inabilities such as lack of experience, knowledge and gut feeling.

The Figure 1.2 indicates the effect of project uncertainties on the accuracy of software size, effort and cost estimates. One may not know the specific nature of the product to be developed to better than a factor of 4 [18]. It is also revealed that alternative estimation approaches failed to provide evidence that their use increase in estimation accuracy [21][23].

It is learnt that expert judgment is most accurate where as formal model is simple to apply, hence these are popular methods for software estimation. Use of software tool to estimate which is calibrated based on past project data is common practice.
Figure 1.2 Software Costing / Effort and Sizing Accuracy versus Phase

(Courtesy : COCOMO II Model Definition Manual )

The challenge with expert judgment method is availability of expert i.e. person having experience of estimating, managing and executing the similar project. The estimate changes with expert or his perception. The Formal model is suitable to apply if rich calibrating data from past projects is available. The expertise is required for use of formal model and selection of input parameters.

1.3 PROBLEM FORMULATION

To begin with, industrial survey has been conducted as an initial activity of this research work. Appendix-A presents detail questionnaire and gist of the responses. Brief analysis of findings are mentioned as below.

- Software Estimation is considered as a highly important task.
• Estimation is done in stages of software Life Cycle termed as Budgetary, Initial, Progressive & Post Construction.

• Project Manager with Technical Expert prepare estimates that are cross verified. knowledge of past project, existing empirical models and software tools are used.

• Post-facto data is maintained as a historical knowledge-base.

• The deviation of estimates with actual values varies by stages but budgetary estimates had variation in the range of 40 – 50 % in later stages range became narrower.

Ranges of deviations of expected estimates by the industry professionals during early and intermediate stages are in line with figures obtained from surveys done by other researchers [18][23]. The finding of this survey formed the basis for proposal of AI models for software estimation.

This thesis addresses the problem of inaccuracies in effort estimates during early and intermediate stages of software development by combining expert judgment with formal models. AI models for software estimation are evaluated to mimic human expertise. AI techniques and technologies such as Semantic Representation and Retrieval, ANFIS are applied with widely accepted formal estimation model COCOMO II.
Overall approach of this study is depicted in Figure 1.3. This model proposed for enterprise where historical knowledge-base including data of past projects has been maintained and rational unified process is adapted for the software development.

Algorithmic Knowledge-based Neuro-Fuzzy Estimator is the set of independent modules enabled as per the stage of software project development. UML based analysis model including the structured document of RFP and design model are input. Semantic representation and matching are applied over RFP & SRS documents to identify similar project. Historical data is used to train the neuro-fuzzy inference system. The human expert interface enables to fine tune result by providing suitable input. Finally, at any time during
software development, effort is calculated to take decisions. The same model can be extended further to estimate cost and schedule.

The Software Estimation Model involving AI technologies viz. Knowledge-base in the form of ontology, Lexical Database Wordnet, ANFIS with Algorithmic Estimation Model COCOMO II has been proposed with following features:

i. Effort estimation with reduced deviation from actual at initial phases

ii. Effort estimation at all phases of software development process

iii. Use of knowledge-base in the form of Ontology to share similar project experience

iv. Provision of Input from Human Expert, Historical Data, UML Analysis & Design Model

v. Support to continuous, dynamic estimation process to reflect changes in input at any time on output parameters to enable generation of alert if possibility of overrun is sensed.

The research is confined to find similar project by semantic matching of RFP, SRS project documents by applying Latent Semantic Indexing, ontology representation and matching with Lexical Database - Wordnet. The experiment is carried to measure size using Function point, Usecase point and Class point method for estimating effort during intermediate stages of software development process. The adaptive neuro-fuzzy inference system integrated with COCOMO II has been evaluated and modified user interface is proposed for more realistic way of input.
1.4 ORGANIZATION OF THESIS

Thesis is divided in chapters and appendices. Chapter 1 introduces software development modalities, role of software estimation, and discusses motivation and problem formulation. The Chapter 2 is Analysis of Estimation Approaches. Review of literature is discussed in this chapter. The literature is classified as software estimation approaches at different stages and underlying AI techniques and technologies. Chapter 3 explains details of experiment conducted for early stage estimation. Semantic matching for software project document is discussed. Chapter 4 depicts experiments conducted and results obtained for intermediate stage estimation. Neuro-fuzzy approach along with COCOMO II is explained in detail. Chapter 5 summarizes conclusion and future scope. Appendix A is consolidated result of survey conducted to know about perception of Indian Software Industry about software estimation. Appendix B presents COCOMO II cost driver’s and scale factor’s mapping from qualitative to quantitative values. Appendix C presents actual effort needed for the projects along with cost driver values. This data has been referred from NASA and COCOMO literature. Next, the publications derived from research work are mentioned. Finally References are listed.