CHAPTER 2
THEORITICAL BACKGROUND

2.1 INTRODUCTION TO COST ESTIMATION

Cost estimation of a software project traditionally starts with quantification of human potential. This estimation is prone to error and tends to introduce inaccuracies that creep into it. The success in Engineering and evaluation of Software depends only when the stages of the SDLC are completed on time. One of the crucial tasks associated with Software Project Management, is estimating cost estimation. Now-a-days software development has become competitive and software developers struggle because of many reasons like delivering product on time at the specified cost ensuring desirable quality. This helps us to understand the importance of estimating effort, schedule in early stages of development of the software. It cannot be concluded whether the exact value of effort can provide accurate schedules though both are closely related with each other. Schedules may slip when vague Requirements / Specifications are to be executed; i.e. when clients make unanticipated change, improper training and personnel availability, sometimes mistakes can happen in initial stages if not corrected in the beginning.

Effort estimates thus computed can be used to derive project plans, financial forecasting such as budgeting, pricing processes, and bidding; it is also used to make iterative plans. In estimation models the inputs supplied are
cost drivers and constraints like financial, manpower, risk factors, etc. In algorithmic cost estimation models the necessary input metrics are provided and not harder to compute on this direct mathematical formula is linked to produce estimated output. By adjusting the input metrics improvement can be made in the estimation.

The view of the estimation process is shown below in Figure 2.1

![Figure 2.1 Estimation Process](image)

The list of resources to be estimated in the iterative cost estimation process of charging includes but it is not limited to:

- Labor
- Materials
- Services
• Software used
• Hardware used
• Infrastructure Facilities
• Contingency Cost

2.2 CONTRIBUTION OF COST ESTIMATION ACCURACY

This emerging research in accurate cost estimation has been carried out for several decades with huge number of researches have to look into the various aspects in the estimation process. This estimation process benefits the appropriateness of estimation certainly contributing in project management arena to provide quality management, scope management, communication management, Risk management, Human resource management, procurement management and Time management.

Currently, none of these models are accepted by the whole community since none of them performs reliable-enough in different environments. We do not have a generalized model that we can apply in different environments of various organizations. Many factors and issues that affect the estimation are not discovered. Some of the reasons of not creating any best model for estimation are:

• Development environment does not remain the same and is evolving continuously
• We are unable to get the measures that reflect the complexity of the system
There are many interrelated factors that affect the project development process. These factors must be discovered and their impact on the effort estimation must be analyzed.

2.2.1 SQM

Software Quality Management aims at managing the software quality in its process of development. This quality refers to user’s satisfaction. The team members should have the quality culture of the industry without making any unnecessary delay and cost overrun because of corresponding delay. Quality management and software project management are different to ensures independence.

How well and fine a software is designed is called as software quality. This design is made conformance in conformance with the specification. It should also have high degree of goodness in terms of delivery of a software product or a software services. This software quality referred is not absolute it is multidimensional construct. It should also fit to the intended purpose (Kitchen ham 1989).

2.2.2 Scope Management

The familiarity of Scope Management is all about what the deliverables should include and what not in the project and also the assigned budget and the projected closure time. Any variation may result in the project failure since it increases budget and duration. When extent of the software project is improperly defined it leads to confusion, cost overrun and incomplete delivery. The Scope about its functional requirement directly reaches from the client interpreted into use cases and non functional
requirements reach from either side of client’s constraints and expert constraints.

2.2.3 Communication Management

Communication management deals with members of team to understand their accountability and roles on the project. If there is no proper understandability exists between developers then such queries may lengthen the duration. Communication is vital between members of the software development team, with service providers and vendors. This communication improves overall success of the project. The software project manager is the sole in-charge for information flow in a timely manner. Several research studies proved that major influential factors relating to software project success are involvement of the user, strategic and executive management and concise information about its requirement. If the above factors are not taken care it leads to threats to the overall success of it.

2.2.4 Risk Management

Project failure can because of lack of risk management (Capers 1994). Successful software systems could be produced if and only if projects are successfully mitigated. The success is directly proportional to risk in software projects because this activity is based on advanced technology. Risk can result in disallowed events like termination or delay.

The key areas of risk areas are

- New and advanced technology
- Requirements of software systems
- Architecture of the system
• Non functional area it adopts.
• Performance of the software system

2.2.5 Human Resource Management

Schedule delay and cost overruns may dominate if team personnel do not possess the skill set to perform the development. Specialized talents are required to perform tasks with recent technologies. Resources are limited now only in software development but for other projects. (Chemuturi, JeanScheid 2010). Greater pressure is mounted on software human resources. Software project manager is held responsible for marshalling the man power resources carefully. Especially human resources are servicing multiple projects in parallel. The different soft skill sets required in development of software projects are programmers, graphic designers, system architects, system integrators and functional experts. Programmers are chosen from the respective platform in which the product has to be developed. Graphic designers are expertise in web page designing and front end designing. Database administrator assists the software programmers in query retrieval and database optimization. The conceptual model definition was developed by the system architects, with a clear picture about structure, behavior and views of the system. The various components developed by the programmers are integrated with an eye on the end product is built in conformance to the specification.

2.2.6 Procurement Management

Procurement management includes plan procurement, conducting, administering and completing procurement successfully. If these are not done properly the external environmental conditions greatly influence the project.
In every organization IT procurement process is either formal or informal. IT procurement is significant because technological evolution is increasing day by day with multiple vendors in the same area. (Robert L Heckman 1994)

2.2.7 Time Management

Accurate estimation of time is a decisive skill in software project. And of course it affects the success of the project. Effective Time management is a process of pre planning and practicing conscious control over time spent on activities to improve productivity and hence accurate estimation of time is very essential. ETM is incredibly vital whereby interruption is hindered in this scenario. ETM is majorly influenced by task scheduling, setting up of goal and proper communication, since it is difficult to synchronize these three. Various methods exists for estimating the time like Bottom-up estimation does breaking of task into detailed task and then estimated. In Top-Down Estimating timeline is computed based on past projects or previous experiences that they have gained. Other estimating methods are comparative three point and parametric.

2.2.8 Objectives of Software Development Effort Estimation

High level resource assessment is necessary to develop a project in a refined manner. Since this key role helps in predicting effort and time span for completion of project. Many researchers declare that without proper planning and unrealistic approaches the industry fails. Having these in the forefront accurate estimation persuades now a days. Effective and strong estimates help the estimators to plan and control successfully. It is principally necessary to estimate effort for software projects and set deadline to do it profitably without cost overruns, schedule slipping and with utmost customer satisfaction. Hence the main objective of estimation does balancing between
the three factors namely cost, schedule and quality. A acceptable mix of all the three or a good relationship of all the three will yield a good software product.

Effort estimation is often done in the early phase (planning) of the project. Following are the steps are followed for estimation purposes:

- Set the objective for the effort estimation
- Make a project plan keeping in mind the available data and resources
- Write down the software requirements
- Work out the complete details of the software systems
- Use multiple effort estimation techniques
- Compare different estimation based on different estimation techniques and iterate the process
- Monitor the project actual cost, and feedback the result to the project management

2.2.9 Estimations in SDLC

The various estimates in software development life cycle are effort, cost, duration and size. This is shown in the Figure 2.2. Effort estimation is useful for proper planning, investment analyses and for fixing the price and to participate in bidding process. Creating conceptual estimate is a major activity of cost estimation. Duration of a project is a measure of elapsed month within which the software project has to be completed. This tentative schedule of completion is mainly based on influencing parameters.
Figure 2.2 Estimates in SDLC
2.3 EFFORT ESTIMATION MODELS

In reality, early estimation and accurate prediction permits the significance of diminishing project risks and can be used to maintain the following

- Decision making process in software development;
- Maintenance and parameters assessment with a concern towards efficiency;
- Cost among different project comparing productivity;
- Resource allocation and manpower allocation etc.

Generally, there exist many methods for software cost estimation, which are divided mainly into two groups: Algorithmic and Non-algorithmic. Using these groups of models estimation can be computed accurately. If the requirements are known better, their performance will be better.

2.3.1 Regression-Based Techniques

The prominent Regression-based techniques used in combination with model- based techniques including functionality based estimation based model, checkpoint, and SLIM models. The classical approach of modeling using least square is the Ordinary Least Square method. The available commercial packages are SPSS, S-Plus 2000, STATA, Gen STAT and Minitab, etc. The response variable is calculated using Equation (2.1).

\[ Y_t = \beta_1 + \beta_2 x_{t2} + \ldots + \beta_k x_{tk} + e_t \]  \hspace{1cm} (2.1)

\( Y_t \) response variable for i th observation.
\( x_{t_2} \ldots x_{t_k} \) is the regression variables or predictor variables for \( t^{th} \) observation

\( \beta_2 \ldots \beta_k \) is the response co-efficient and \( \beta_1 \) is an intercept parameter

OLS also called as linear least squares method works using a linear regression model. The sum of squared vertical distances in the observed data set and in the predicted linear approximations is minimized. The OLS method is of choice when the degree of freedom is available in surplus and no data is found to be missing and there is no misunderstanding. OLS Method also violates the assumptions hence the predictor variables are not expected to be correlated. An improvement in standard OLS is robust regression. This is an improved method in effort estimation. In certain category of robust regression methods like Least Median Square is preferred. Some regression methods which are exist use datapoints with two to three standard deviations.

2.4 THE ALGORITHMIC COST ESTIMATION MODELS

Algorithmic models are also called as parametric cost estimation models which use a mathematical model they are usually embodied as popular technique which deals with prior experience and are less inclined when compared with thoughts of a human. These estimates are purely based on inputs and facts. Different variables that affect the estimation are considered. These variables are often called as cost drivers. Many quantitative software effort estimation models have been developed. They are mainly categorized into the following categories Analytical Models and Empirical Models. An analytical model uses data from the previous project in order to evaluate the current project. An empirical model is based on some global assumption. It uses global assumptions to estimate the effort required in the completion of the project. Some of the global assumptions are developer’s rate to solve the problems and number of problem available.
2.4.1 SEER

SEER (System Evaluation and Estimation of Resources) is a proprietary model owned by Galorath Associates, Inc. In 1988, Galorath Incorporated this work on the original version of SEER-SEM which resulted in an initial solution of 22,000 LOC. SEER is an algorithmic software project management application planned specifically for effort estimation. This model is relied on the initial effort of Dr. Randall Jensen. The mathematical equations that are used in SEER were not available to the public, but Dr. Jensen in his writings made the basic equations available for review. The basic equation of Dr. Jensen is also referred as software Equation (2.2).

\[ Se = Cte(Ktd)0.5 \]  (2.2)

where, ‘S’ is the effective LOC, ‘ct’ is the effective developer technology constant, ‘k’ is the total life cycle cost in man-years, and ‘td’ is the development time in years.

2.4.2 SEER-SEM model

SEER-SEM (Software Estimation Model) is an influential software tool for estimating development and maintenance cost of a software. It also includes software project management constraints like labor rates, schedule considerations, reliability and risk as a function of size, technology. This is traditional planning software which can do tracking of software, cost management and it can also be referred as optimizer in software planning processes. This is also referred as multi-faceted approach which leverages software industry and the historical data available with proven formulaic approach. This is a windows based interface for describing the software projects.
2.4.3 Top-down estimating method – SLIM

Software LIfe Cycle Management (Putman’s SLIM) is a programmed macro estimation model for software cost estimation based on the Norden / Rayleigh function. Also called as top down estimation model, with the global properties cost estimates are found and then projects is subdivided into low level local components. Since minimum details are required it is easier and faster.

It uses linear programming, statistical simulation, program evaluation and review techniques to obtain a software cost estimate. SLIM enables a software cost estimator to perform the functions like Calibration, Building and Software sizing. The estimates are very sensitive to the technology factors extremely. This macro estimation model is used to perform fine tuning by interpreting the database information usually called as calibration.

The parameters used in these models are very less but it is not usually preferred for smaller projects.

The effort is computed using Equation (2.3).

\[ K = \left( \frac{\text{LOC}}{(C \cdot t^{4/3})} \right) \cdot 3 \]  

\(2.3\)

‘K’ is the life cycle effort in working years totally, ‘t’ is development time and ‘C’ is the technology constant which can range from 610 to 57314. This technology constant is extremely sensitive and derived by combining the effect of using software tools; programming languages, research methodology and quality assurance (QA).
2.4.4 PRICE-S

PRICE – S was developed by Martin Marietta originally called RCA Price, subsequently referred GE Price (F. Freiman and Dr. R. Park 1977). The primary input of PRICE - S is Lines of Code. This model consists of three sub models. The glimpses about its functionality are listed below. They are namely acquisition sub model, sizing model and life cycle cost sub model. The output of this model is estimated in person hours and months. The other optional outputs of these models are Gantt charts, sensitivity matrices, resource utilization profiles and scheduling reports. This model does calculation of cost output in three phases namely enhancement, growth and maintenance.

2.4.4.1 Acquisition Sub Model

This acquisition submodel predicts costs of the software and its schedules. This covers all types of projects, including business software projects, communication software, commanding and control, avionics and space systems. PRICE-S provides solutions to problems like reusability, reengineering, rapid modeling, OO development and better productivity.

2.4.4.2 Sizing Sub Model

This sizing submodel makes easy estimation on the software size which is to be developed. The metrics associated with sizing is SLOC, FP and POP’s. POP (Minkiewicz 1998) was purely object oriented. This is based on the earlier research done by Chidamber on OO system. [Chidamber and Kemerer 1994; Henderson-Sellers 1996].
2.4.4.3 Life-Cycle Cost Sub Model

The Life cycle cost submodel is used for speedy and premature costing of the maintenance and support phase for the software. This model is used in conjunction with the earlier Acquisition Submodel, which does the development costs and design parameters estimation. Rigorous analysis is facilitated by Monte carlo simulation utility which is available in this model. The latest update in this PRICE Systems is Foresight 2.0 for predicting the cost, effort and time for commercial applications.

2.4.5 Proxy Based Estimation

PROBE estimation was developed by Watts Humphrey (of the Software Engineering Institute at Carnegie Mellon University) for software effort estimation. Engineers who involved themselves in software project they themselves decide personally. PROBE works in a fashion that engineer who built the software previously predicts the same effort based on the projects realized earlier. The information about the projects developed and the amount of effort involved etc. are recorded in a table and this database supports in estimation by breaking individual component or modules and its corresponding effort summing up of all these produces the requisite effort.

2.4.6 Function Point Analysis

Function Point Analysis (A. J. Albrecht and J. E. Gaffney 1983) can be applied to software effort estimation. Basically function points are a standard unit of measure which represents the functional size of a application. The ability of this model is to accurately estimate its project cost, project duration and project staffing size. A perceptive of other vital metrics, such as Cost per FP, FP's per hour and the productivity profits of using innovative or
different tools. The functional size imitates the quantity of functionality that is appropriate to and recognized by the user in the software business. It is mainly independent of the technology used to implement the software.

FPA is mainly based on the requirements of the user. This is user centric and platform independent. The goal of this estimation method is to provide an estimation schema based on functional size of the software. Effort is one of the valuable assets to the people in strategic level for planning and management activities. This decision making is a software related process which supports resource and staff allocation by breaking a larger task system into smaller components which can be analyzed and understood in an efficient way. This functional decomposition is done in a structured manner algorithmically. User’s design is collected based on this logical design and from the functional viewpoint acceptance testing is performed.

Three major capabilities for supporting the whole SDLC are estimation, measurement and assessment.

**Estimation**

Checkpoint forecast effort at the following levels of granularity namely project, phase, activity, and task. Estimates also include end products or deliverables, development defects, development costs, resources and schedules.

**Measurement**

Checkpoint helps users to incarcerate project metrics to carry out benchmark analysis, recognize the best hygienic practices, and to develop internal estimation knowledge Templates.
Assessment

Comparison of actual and estimated is facilitated using check points in the light of various industrial standards included in the knowledge base. Process improvement if required could be made to assess the benefits of implementation.

2.4.7 SPQR

Software Productivity, Quality And Reliability Model - 20 (Jones 1987) is purely dependent on the past project database. It uses FPA for sizing the volume of a program. SPQR – 20 is the twenty different questions that are taken into consideration in this model. This software estimation package is available commercially and provides excellent effort. Accuracy of the cost estimate prior to definition of requirements is fair. This model is easier to use. This model also does forecasting of development and maintenance cost. There are four commercial versions in it namely SPQR 10, 20, 50 and 100. This indicates the level of refinement in it.

The essential functions in FP are

Inputs – Set of input data supplied by other programs and the users.

Outputs – Set of output data produced by users

Inquiries – interviewing the users means of the software system

Data files – Information is in a collection of database records for easy accessing and alteration.

Interface – Database and file sharing with other systems.
2.4.8 Bottom - Up Estimating Method - COCOMO

WBS provides a structure with hierarchy wherein the activities of work need to be performed inorder to complete the project within its scope. WBS decomposes into smaller components and manageable segments of pieces referred as sub components. Independently all the tasks are estimated. Together these individuals are combined to produce estimate belonging to a single project. This method is a painstaking method whose accuracy mainly depends on correct Work breakdown structure. This resource intensive approach assumes all team members to have a good back ground knowledge and understanding on all the project specifications and requirements. Then only the broken tasks could be integrated at various levels. This is very time consuming because the level of identification of knowledge is to be complete for breaking down and integrating it.

COCOMO (Constructive Cost Model) is an empirical estimation scheme proposed in 1981. It is a model for estimating effort, cost, and schedule for development of software projects. This parametric model certainly provides better results. Barry Boehm definition towards his model "Basic COCOMO is good for rough order of magnitude estimates of software costs, but its accuracy is necessarily limited because of its lack of factors to account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other project attributes known to have a significant influence on costs."

Constructive Cost Model is an open model, the details it includes are

- The basic cost estimation equations
- Every assumption made in the model (e.g. "the project will enjoy good management")
Every definition (e.g. “the precise definition of the Product Design phase of a project”)

The estimation costs incorporated into considerations are explicitly stated (e.g. project managers and technical people are included, private secretaries etc. aren't)

COCOMO is a Size- based models are regarded based on project size in LOC or KLOC. The COCOMO estimates are more objective and repeatable than estimates made by methods relying on other proprietary models. COCOMO can be calibrated to reflect the development environment of software, and it can produce estimates.

In COCOMO 81, effort is expressed as Person Months (PM) and it can be computed using Equation (2.4).

\[ PM = a \times \text{Size}^b \times \prod_{i=1}^{15} EM_i \]  

(2.4)

where “a” and “b” are the domain constants involved in the estimation model. EM is the effort multipliers. This estimation scheme accounts the experience and data of the past projects, which is extremely complex to understand and apply the same.

2.4.8.1 Categorization of COCOMO Suite

COCOMO has progressed to meet the needs of the user since the complexity is more. All these derivative models are similar in nature but are customized for different software development situations they are Constructive Incremental COCOMO (COINCOMO 2004) and Database
doing business as COCOMO (DBA COCOMO 2004). Constructive Cost Model II (COCOMO II 2000) estimates the software effort analyses software requirements design, implementation, and testing. Constructive systems Engineering Cost Model (COSYSMO 2002) estimates the system engineering effort associated with the software, its design, implementation, and testing. Informative dependability Attribute value Estimation (i - DAVE 2003).

Constructive Security Cost Model is developed for secure costing (Colbert et. al. 2006) and is based on behavior- analysis. COSMIC (COSMIC 2007) provides a consistent method of effort estimating a functional size of software from the functional domains commonly referred to as Management Information Science software and other real-time software. COSMIC Model is purely based in rules and underlying principles to the FUR of the software.

Constructive System-of-Systems Integration Cost Model estimates effort linked with the integration of software system components (COSoSIMO 2004). The COCOTS model predicts the lifetime cost associated with assessment, tailoring and Integration. The process by which COTS components are selected for its use in the larger system being developed is termed as Assessment. Tailoring refers to those tasks to prepare COTS program regardless of the system incorporated. Constructive Rapid Application Development Model (CORADMO 1999) is also in the same flavor of COCOMO II focusing in development cost using Rapid Application Development techniques. Constructive Product Line Investment Model (COPLIMO 2003) is purely based on its relative cost of reuse which includes additional cost of writing reuse. CONstructive QUALity MOdel (CoQualMo) focuses on quality cost projected in approximation cost prior to the development period it also adds defect removal.
COCOMO II has types of models

- Early Design Model (EDM)
- Post-Architecture Model (PAM).

The EDM is preferable in the initial development stages where detailed information is not available. This information includes actual software and its overview about developed process especially entire architecture. In other words EDM is a high-level model which is used to see the sights of incremental development strategies.

PAM works with in-depth requirement and the software is ready for its development. Hence this can be referred to develop and continue with a fielded system. The PAM ought to have a life-cycle architecture package, which has detailed information on metrics like cost drivers and enables to find more enhanced cost estimates. EDM and PAM use same fashion for product sizing and scale factors. The PAM model has 17 Effort multipliers and 5 scale factors. Cost drives have a rating level that expresses the impact of the driver on development effort, PM. The popular empirical estimation models like SLIM, COCOMO, SEER - SEM influenced by Putnam’s framework, Price-S, SLIM and other models require accurate inputs for estimation of effort and the output is mainly independent on the input. The COCOMO model is a pragmatic model that was derived empirically based on the data from large number of software projects.

In the software development process, cost prediction is generally computed using the Equation (2.5)

\[ PM = A \times (\sum \text{Size})^B \times \Pi(EM) \]  

(2.5)
In the above equation PM refers to Person Months, A is a calibration factor, Size refers to functional size of a software module, B has a non linear effect in estimation and is termed as scale factor and EM is the influential effort multiplier.

The various COCOMO models are listed in the Table 2.1

**Table 2.1 Various Model of COCOMO**

<table>
<thead>
<tr>
<th>Model</th>
<th>Scope of the Model</th>
<th>No. of Additive Factors</th>
<th>No. of Exponential Factors</th>
<th>No. of Multiplicative Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCOMO</td>
<td>Effort Estimation and Scheduling</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>COCOMO II</td>
<td>Effort Estimation and Scheduling</td>
<td>1</td>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>COSYSMO</td>
<td>Effort estimation</td>
<td>1</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>COCOTTS</td>
<td>Assessment customization and Integration effort</td>
<td>3</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>i - DAVE</td>
<td>Information dependability attribute</td>
<td>1</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>CoRADMo</td>
<td>Estimates the schedule and personnel effort</td>
<td>1</td>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>COPLIMO</td>
<td>Estimates the schedule and personnel effort including reuse of code.</td>
<td>1</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>CoQualMo</td>
<td>Approximation cost prior to the development period and defect removal</td>
<td>3</td>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>
2.4.8.2 Scale factors in COCOMO II

This scale factor does variation in software effort exponentially hence selection of scale drivers is done with proper justification or on some principle. Each scale factor has 6 levels of evaluation namely Very low, Low, Normal, High, Very high and extremely high. The specific value is called scale factor.

Table 2.2 Scale factors in COCOMO

<table>
<thead>
<tr>
<th>Scale Factor</th>
<th>Explanation</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF 1</td>
<td>Precedentness</td>
<td>PREC</td>
</tr>
<tr>
<td>SF 2</td>
<td>Development / Flexibility</td>
<td>FLEX</td>
</tr>
<tr>
<td>SF 3</td>
<td>Architecture / Risk resolution</td>
<td>RESL</td>
</tr>
<tr>
<td>SF 4</td>
<td>Team cohesion</td>
<td>TEAM</td>
</tr>
<tr>
<td>SF 5</td>
<td>Process maturity</td>
<td>PMAT</td>
</tr>
</tbody>
</table>

Earlier studies have offered imminent into accuracy in estimates, functionality being delivered and success on project manager’s appraisal on importance of precision. The major contributor to projects success is a good accurate estimate (Berntsson-Svensson, Aurum, Procaccino, Verner et al 2005). The accuracy in effort estimation has highly increases the success possibility of the project (Brooks, I. D. Steiner 1972). Staying within budget and schedule was ensured instrumentally victorious and it is also difficult to be successful in it. (Verner, Evanco et al 2001.)

The Cost drivers in COCOMO II effort equation produce the EMi values. The four cost drivers are product attributes, platform attributes, personnel attributes and project attributes.
### Table 2.3 Cost drivers of COCOMO II

<table>
<thead>
<tr>
<th>Cost drivers / Attributes</th>
<th>Effort Multiplier</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Required software reliability</td>
<td>RELY</td>
</tr>
<tr>
<td>Product</td>
<td>Size of application database</td>
<td>DATA</td>
</tr>
<tr>
<td>Product</td>
<td>Complexity of the product</td>
<td>CPLX</td>
</tr>
<tr>
<td>Product</td>
<td>Required Reusability</td>
<td>RUSE</td>
</tr>
<tr>
<td>Product</td>
<td>Documentation Match to life cycle needs / demand</td>
<td>DOCU</td>
</tr>
<tr>
<td>Platform</td>
<td>Execution Time Constraint</td>
<td>TIME</td>
</tr>
<tr>
<td>Platform</td>
<td>Main storage Constraint</td>
<td>STOR</td>
</tr>
<tr>
<td>Platform</td>
<td>Platform volatility</td>
<td>PVOL</td>
</tr>
<tr>
<td>Personal</td>
<td>Analyst Capability</td>
<td>ACAP</td>
</tr>
<tr>
<td>Personal</td>
<td>Programmer Capability</td>
<td>PCAP</td>
</tr>
<tr>
<td>Personal</td>
<td>Application Experience</td>
<td>APEX</td>
</tr>
<tr>
<td>Personal</td>
<td>Platform Experience</td>
<td>PLEX</td>
</tr>
<tr>
<td>Personal</td>
<td>Language and Tool Experience</td>
<td>LTEX</td>
</tr>
<tr>
<td>Personal</td>
<td>Personnel Continuity</td>
<td>PCON</td>
</tr>
<tr>
<td>Project</td>
<td>Use of Software Tools</td>
<td>TOOL</td>
</tr>
<tr>
<td>Project</td>
<td>Multisite Development</td>
<td>SITE</td>
</tr>
<tr>
<td>Project</td>
<td>Required Development Schedule</td>
<td>SCED</td>
</tr>
</tbody>
</table>

Cost drives that are listed above in the table are used to capture characteristics of the software development that affect the effort to complete the project. Cost drives have a rating level that expresses the impact of the driver on development effort, PM. For the purpose of quantitative analysis, each rating level of each cost driver has a weight associated with it. The
weight is called Effort Multiplier. The average EM assigned to a cost driver is 1.0 and the rating level associated with that weight is called Nominal.

In COCOMO II, the effort requirement can be calculated using equation (2.6,2.7)

\[ PM = a \times Size^E \times \prod_{i=1}^{17} EM_i \]  \hspace{1cm} (2.6)

where

\[ E = B + 0.01 \times \sum_{j=1}^{5} SF_j \]  \hspace{1cm} (2.7)

COCOMO II is related with 31 factors; LOC is measure as the estimation variable, 17 cost drives, 5 scale factors, 3 adaptation percentage of modification, 3 adaptation cost drives and requirements & volatility.

Determining size of a project is challenging for an estimation model. Software projects are composed of new code, reused code from some other projects with slight modification or without even modifying. COCOMO II uses size data that pressurize effort which is a new line of code and this code is modified and not taken as such from other projects. The code that is copied from similar projects and slightly modified is also counted as new LOC.

2.4.8.3 Product Factors

The characteristics of a product deeply influence development effort of software. A Complex product definitely requires more reliability requirement. Working with a bulk data and complex relationship in a database etc these are termed as product factors.
RELY

Required software reliability refers to the extent to which software performs its intended function. If the failure of software creates risk and inconvenient to human then the rating is very high else it is low.

DATA

Product development environment should attempt to capture huge data requirements. If the database size available is larger definitely with more test data calculating becomes easier.

CPLX

Product Complexity covers several areas namely computational, control, data management, device dependent and user interface management. The complexity is the cumulative weights of the above operations.

RUSE

Required reuse refers to recycling of same code or similar constructs requiring prevalent testing and detailed documentation. These make certainly usage of the identical in other applications.

DOCU

Documentation and match to life cycle needs includes proper catalogs with data dictionary and its suitability in terms of life cycle needs.
2.4.8.4 Platform Factors

PVOL

Platform volatility refers to hardware and software complexity. This is more because of technological developments day by day we find technocrats change the world. Adaptability and background knowledge to such altering situations affect the rating from low to very high.

TIME

Execution time constraint obligatory in the lead of a software system. The rate of consumption of resources utilized by the system and its subsystem percentage is TIME.

STOR

Main storage constraint refers to constraints variables imposed on software system and subsystem. This rating mainly ranges from nominal to extra high.

2.4.8.5 Personnel Factors

This has a stronger influence on the effort directly and there exists an over confidence in performance productivity.

ACAP

Analyst capability refers to the personnel who work with the requirements, detailed design and high level design in bonding with the project. The experience gained by the programmer should be taken into consideration. It is should be rated with APEX.
PCAP

Programmer capability was not based on assessment of individuals rather based on the team effort. When individuals work in a team they should be able to communicate among themselves cordially and also cooperation is expected. If the team can perform better then it pressurize more productivity.

APEX

Application Experience refers to knowledge subjectivity of the developer towards a particular application which he is supposed to develop. Whether he / she has developed similar application elsewhere. A very low rating refers to lesser experience in terms of months and Very high refers to several years of experience. Rating Extra is high is not applicable.

PLEX

Platform Experience influences software productivity in actual fact. Understanding and knowledge gained on specific platform like GUI, cross platform, middleware technology, database, distributed computing capabilities and networking is termed as PEXP.

LTEX

Language and Tool Experience is the quantification of experience gained by the members in the project team towards programming language and development specific software tools. A low rating means very low referred in terms of meager experience gained by the software professional. A
very high measure of rating is given for several years of experience namely five plus years.

**PCON**

Personnel Continuity is expressed in terms of the software project’s yearly personnel turnover rating level from very high to very low.

2.4.8.6 Two – Stage Estimation Schemes of COCOMO

From the above discussions, it is observed that the COCOMO models uses two stage estimation design for finding the effort requirement and it can be illustrated as:

**CASE-1**

In the Figure 2.3(a), let the shaded area be the effort required for a given LOC. Let LOC = 100, then by assigning the nominal values to all effort multipliers, the required effort in the semi detached mode can be calculated as \((a\times \text{size } b)\). i.e. \(3\times 1001.12 = 521.3\) Person Months.

**CASE-2**

Considering a projected value for ACAP (Application capability) effort multiplier and the values of all other Effort Multipliers are nominal, then for the same LOC, the required effort can be calculated as \(3\times 1001.12\times 1.46 = 761.15\) Person months. This is illustrated in the Figure 2.3(b).
CASE-3

In this case, a projected value is considered for CPLX (Complexity) only and all other Effort Multipliers are kept at nominal level, then for the same LOC, the required effort can be calculated as $3 \times 1001.12 \times 0.7 = 364.93$ Person Months, which is described in the Figure 2.3(c).

Hence, in principle, the effort is calculated by multiplying the estimation variable with the constant ‘a’ in the first stage, and some effort can be added with or deducted from the calculated effort at the second stage. In addition to that, each user should calibrate the model and the attribute values in accordance to their own historical projects data, which will reflect local circumstances that greatly influences accuracy of the model. From these viewpoint, whenever using the algorithmic effort estimation models, it is preferred that the impacts of cost drives have to be quantified and assessed in a proper way using Table 2.4.

**Figure 2.3 Two-Stage estimation of COCOMO**

<table>
<thead>
<tr>
<th>Multiplication Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal</td>
<td></td>
</tr>
<tr>
<td>Additive</td>
<td>Multiplication factor = 1.46</td>
</tr>
<tr>
<td>Deductive</td>
<td>Multiplication factor = 0.7</td>
</tr>
</tbody>
</table>

2.3(a) 2.3(b) 2.3(c)
Table 2.4 Range Values of cost drivers

<table>
<thead>
<tr>
<th>Cost Drivers</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>0.82-1.26</td>
</tr>
<tr>
<td>DATA</td>
<td>0.90-1.28</td>
</tr>
<tr>
<td>CPLX</td>
<td>0.73-1.74</td>
</tr>
<tr>
<td>RUSE</td>
<td>0.95-1.24</td>
</tr>
<tr>
<td>DOCU</td>
<td>0.81-1.23</td>
</tr>
<tr>
<td>TIME</td>
<td>1.00-1.63</td>
</tr>
<tr>
<td>STOR</td>
<td>1.00-1.46</td>
</tr>
<tr>
<td>PVOL</td>
<td>0.87-1.30</td>
</tr>
<tr>
<td>ACAP</td>
<td>1.42-0.71</td>
</tr>
<tr>
<td>PCAP</td>
<td>1.34-0.76</td>
</tr>
<tr>
<td>PCON</td>
<td>1.29-0.81</td>
</tr>
<tr>
<td>APEX</td>
<td>1.22-0.81</td>
</tr>
<tr>
<td>PLEX</td>
<td>1.19-0.85</td>
</tr>
<tr>
<td>LTEX</td>
<td>1.20-0.84</td>
</tr>
<tr>
<td>TOOL</td>
<td>1.17-0.78</td>
</tr>
<tr>
<td>SITE</td>
<td>1.22-0.80</td>
</tr>
<tr>
<td>SCED</td>
<td>1.43-1.00</td>
</tr>
</tbody>
</table>

This model is time consuming and it may be difficult to overlook with the system cost in early phase of its development. Since the significance of the vagueness and uncertainty features that are inhabited in the effort estimation drives due to the cognitive judgments are less, this Imputation approach can be preferred and applied to change the estimation scheme of the COCOMO II, which can substitute values in the place of vague information.
2.5 THE NON-ALGORITHMIC COST ESTIMATION MODELS

Non algorithmic model estimates not by mathematical formulas but by means of human intelligence. These models they even work with partial inputs i.e. even when subset values of certain inputs are available. Differing algorithmic models it does based on inferences and analytical similarity.

2.5.1 Effort estimation by analogy

A straightforward method of estimation is analogy method. The two well-liked models are ESTOR (Mukhopadhyay 1992) and ANGEL (Shepperd M., Jorgensen M. 2007). ESTOR is a case based reasoning model. Input metrics are FP values. Analogous projects are identified and taken as initial solution. Then it is tuned. ANGEL is a generalized approach where analogue projects are neighbors and vector distance are calculated; several analogous data are taken into account. The user can involve in deciding the optimum metrics. Effort relating to the new projects is estimated based on the mean effort value with reference to the neighboring projects. The distinction between ESTOR and ANGEL lies in number of analogues.

- Case-base reasoning model has 5 basic processes:
- Characterizing the proposed project.
- Target case specification;
- Search for ample of cases to serve as unique analogy;
- Reassigning of solution from the source case to target;
- Find the deviation between original and target;
- Fine-tuning of initial solutions based on the deviations observed.
This method uses information from similar projects and derive estimate based on the past features of data, because it is a similar form of reckoning as human problem solving capability (Angelis and Stamelos 2000; Leung 2002). This method finds the similarities between projects it can be used in the early phase of the project. It is an easy way of finding effort of the project. The major problem with existing analogy-based technique is limited because of their inability to handle with non-quantitative data and missing values. Analogy based method shows best results in 60% of the cases and 30% of the cases fail because of worst predicted accuracy, hence suggesting some instability (Ruhe et al 2003). It can be a promising in majority cases but fails in certain cases.

The similarities between functions are identified by Euclidean similarity (ES) and Manhattan similarity (MS) (Shepperd and Schofield 1997). The similarity computation is done as follows using Equation (2.8,2.9).

$$Sim(p,p') = \frac{1}{\sqrt{\sum_{i=1}^{n} w_i Dis(f_i,f_i')}} + \delta$$

$$\delta = 0.0001$$

(2.8)

$$Dis(f_i,f'_i) = \begin{cases} 
(f_i - f'_i)^2, & \text{if } f_i \text{ and } f'_i \text{ are numeric or ordinal} \\
1, & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\
0, & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i \neq f'_i 
\end{cases}$$

(2.9)

where ‘p’ and ‘p’ are the projects, ‘wi’ is the weight assigned to each feature and it can vary between 0 and 1. ‘Fi’ and ‘fi’ displays the ‘ith’ feature of a project and ‘n’ demonstrates the number of features. ‘δ’ is used to obtain non zero results. The MS formula is similar to the ES but it does computation by finding the absolute difference between the features.
The similarity function is given below in Equation (2.10,2.11)

\[
Sim(p, p') = 1 / \left( \sum_{i=1}^{n} w_i \text{Dis}(f_i, f'_i) + \delta \right) \delta = 0.0001
\]  

(2.10)

\[
\text{Dis}(f_i, f'_i) = \begin{cases} 
|f_i - f'_i| & \text{if } f_i \text{ and } f'_i \text{ are numeric or ordinal} \\
1 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\
0 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i \neq f'_i 
\end{cases}
\]  

(2.11)

After selecting the K most similar projects, it is made possible to estimate the effort and cost of the new project as per the feature selected. The most common solution functions are: the closest analogy as most similar project (Walkerden and Jeffery 1999), the average of most similar projects (Shepperd and Schofield 1997), the median of most similar projects (Angelis and Stamelos 2000) and the inverse distance weighted mean (Kadoda et al. 2000). The mean describes the average of the effort of K most similar projects, where K > 1. The median describes the median of the effort of K most similar projects, where K>2. The inverse distance weighted mean adjusts the portion of each project in estimation by using the Equation (2.12)

\[
C_p = \frac{\sum_{k=1}^{K} \frac{\text{Sim}(p,p_k)}{\sum_{i=1}^{n} \text{Sim}(p,p_i)} C_{p_k}}{\sum_{k=1}^{K} \frac{\text{Sim}(p,p_k)}{\sum_{i=1}^{n} \text{Sim}(p,p_i)}}
\]

(2.12)

where p shows the new project, p_k illustrates the k^{th} most similar project, C_{p_k} is the effort value of the k^{th} most similar project pk, Sim (p, p_k) is the similarity between projects p_k and p and K is the total number of most similar projects.

2.5.2 Expert estimation techniques

This estimation schema was developed by Barry Boehm (Software Engineering Economics, Boehm 1980); each group member in the estimation team provides an anonymous estimate, the coordinator assesses the estimates
given by the team and combines the estimate. If the estimates are crazy reassessment is done finally effort is computed. Delphi Technique obtains opinion of experts without collectively bringing all of them face to face. The panel members can stay in any geographical area but questionnaires will reach them via computerized systems or Electronic mean. In a conventional committee there is a possibility of biased feedback.

The steps involved in Delphi estimating method are:

1. Specification and estimation form is given to each Coordinator who is part of the cost estimation team.

2. Coordinator calls for a group meeting where in the experts has a brainstorming discussion on estimation issues with the coordinator and others in the team.

3. With autonomy experts fill out the forms secretly without revealing to others.

4. An iteration form is used to record the observations made by the experts all these summarizing activity is taken care by the Coordinator.

5. The Coordinator arranges for group meet, specially centered on these agenda.

6. An expert fills out the forms, again secretly and identity is not exposed even after successful completion, and repeats steps 4 to 6 till appropriate estimates are reached.

A slight flavor of this is Mini Delphi where in face to face meetings are also encouraged. It is also called as Estimate – Talk- Estimate. This concludes that always structured group judgments are far better than
individual assumptions. Delphi method caters the need only in single
dimension and hence it cannot be involved for performing complex
forecasting with multiple influencing factors.

2.5.3 Learning Oriented Estimation

The Artificial Neural Network embarked its evolution in 1943
(Warren S. McCulloch, Walter Pitts) published "A logical calculus of the
ideas immanent in nervous activity" which gave its birth to new era of
intelligence. Machine learning models are currently used in wide range of
applications like financial applications, image processing, voice recognition
system and forecasting applications. Alternate to Least square regression
models ANN is commonly preferred (Gray and MacDonell 1996) is shown in
Figure 2.4.

ANN is a computational simulation of biological neuron. These
models mimic the actual life behavior of neuron. ANN is adaptive model. In
general ANN models are superior in estimation it has an ability to learn from
prior data. It can also generalize from the training pair enabling us to estimate
results in an acceptable manner. The significant challenge faced by the
estimating community is effective prediction of resources. ANN predictive
models where built to tackle down because they are able to model non linear
relationship.

They consider ANNs as promising techniques to build predictive
models, because they are capable of modeling non linear relationships. This
architecture of biological neural networks comprises of interconnected
artificial neuron units. The input layer and the other hidden layers in the MLP
do computation on weighted sum of inputs and generate outputs. Initially
ANN is assigned with random weights and training gradually learns the
complex relationships. The primary function of learning is to train the neural network to perform task independently. Types of learning includes supervised, unsupervised and reinforcement. This neural network has to be trained (Hebbian 1949). Training a network of BDN’s was achieved with the support from synaptic strengths. Since this a most essential choice in an ANN enough care has to be taken in choosing this with respect to the architecture. Supervised learning is preferred if the end result or target is known if the correct outcome is not known then unsupervised or reinforcement is preferred.

Reinforcement learning is analogous to supervised learning (Barto 1983), except that, instead of being present with the target output for each network input, the method provides only a grade. The score is a measure of the network performance over some sequence of inputs. It is suitable for control system applications. In unsupervised learning target outputs are not given. The weights and bias are adjusted to get the results. This is useful for applications like estimation, vector quantization etc.
Input Layer  Hidden Layer  Output Layer

Input #1  
Input #2  
Input #3  
Input #4  
Output

**Figure 2.4 Estimation of effort using ANN**

The types of ANN are Single-layer feed forward network and Multilayer neural network also called multilayer perceptron. Neural network has been worldwide in ongoing research ranging from basic algorithms to efficient learning. ANN finds finer because it can perform tasks which a linear program cannot do. The major disadvantage associated with this is it requires high processing time for large problem and more training.

### 2.5.3.1 Fuzzy decision systems

This decision support systems are based on fuzzy logic, FL is a problem-solving methodology. Zadeh introduced Fuzzy Logic and sets which provides representation schema and modeling it mathematically. This reproduces the human reasoning with a rule base. FL helps us by providing a simple method to derive a definite conclusion even with vague, noisy,
ambiguous and imprecise inputs. It will also survive even with missing data. Fuzzy logic approach controls and mimics like a person in decisions at a faster rate. A fuzzy set is characterized by a membership function, which associates with each point in the fuzzy set a real number in the interval [0, 1], called the degree of membership or grade of membership. These models enhance the ability of the user to interpret the model in their view and to evaluate, criticize and adapt the same (Idri & Adan 2001). Fuzzy models are easy to understand when compared with neural and regression models.

The real difficulty lies in making precise and clear cut decisions. The expression used is called as Fuzzy logic. The four stages in FL are

Stage 1 : Fuzzification generates trapezoidal numbers for the linguistic terms.

Stage 2 : Broadening the complexity matrix by a new linguistic terminology.

Stage 3 : Finding the productivity rate and effort for new terms.

Stage 4 : Defuzzification is the effort required to complete a job.

Finally comparison is done with the existing approach.

2.5.3.2 Genetic approach based decision systems

Genetic Algorithm (GA) is a population-based algorithm used to solve the combinatorial optimization problems (Adriano L.I. Oliveira 2010). Genetic Algorithm inspired by biological evolution is an AI procedure. This is an iterative method in which the population size is constant where the solution is eliminated from each of the individuals in the given population. This is purely based on theory of usual selection and evolution. The end outcome will
be an optimized result based on evolutionary approach. GA are useful for several applications like State Assignment Problem, in the field of Economics, Computer-Aided Design, Chemical kinetics, scheduling applications, game theory, marketing mix analysis, software engineering, stochastic optimization, filtering and signal processing, evolvable hardware design, Learning behavior of Robot using genetic algorithms, Learning fuzzy rule base using GA, Linguistic analysis, which includes grammar induction and other features of NLP like word sense disambiguation, Marketing mix analysis, etc. GA is used for finding reliable and precise estimates. Several researches have proved that GA’s are suitable for estimation.

The algorithm for GA is as follows

**Step 1: Initialization**

For each starting from one, randomly generate ‘N’ solutions which are referred as initial population, P1. Then the fitness is evaluated.

**Step 2: Crossover**

Many cross over techniques are existing One-point crossover, Two-point crossover, Cut and splice cross over, Uniform Crossover, Half Uniform Crossover and Three parent crossover.

Generating offspring for population Qt is:

2.1. Choose two solutions x and y from Pt based on the fitness values.

2.2. Using a crossover operator, generate offspring and add them to Qt.
**Step 3: Mutation**

Mutate each solution with a predefined mutation rate.

**Step 4: Fitness assignment and feasibility assignment.**

Based on objective function, fitness value is evaluated and assigned, Check for its feasibility.

**Step 5: Selection**

Select ‘N’ solutions from the offspring of population, Qt based on their fitness and copy them to Pt+1.

**Step 6: If the terminating criterion is satisfied, stop the search and move back to the current population, else, increment t by 1 go to Step 2.**

To solve complex problems several classes of algorithms exists namely probabilistic, deterministic and stochastic. GA’s are stochastic in nature which is preferred for several branches of engineering and computer problems.

### 2.5.4 Decision Tree Models

Data mining (Knowledge Discovery in Databases (KDD)) is the process of discovering meaningful patterns in huge databases. In addition, it is also has an application that can provide significant competitive advantages to make right decisions. Decision tree is one of common data mining methodologies that provide the both classification and predictive functions simultaneously. A decision tree is a tree whose internal nodes can be taken as tests and whose leaf nodes can be taken as categories. These tests are filtered
down through the tree to get the right output to the input pattern. The advantage of using decision trees here was that it does not require any statistical knowledge. A decision tree is the most widely used tool for decision making. To accomplish this one initially draw a decision tree with different branches and leaf nodes. These branches and leaves should point to all the various factors concerning a particular situation. A decision tree is almost like a decision support tool. It uses a tree-like graph of decisions and their possible outcomes which include resource costs, event outcomes, and utility.

2.5.4.1 Decision stump

Decision stump (Guo peng yang, Xin zhou, Xuchu yu 2009) is a decision tree with one internal node (root) which is immediately connected to the terminal nodes. It makes a prediction based on the value of just a single input feature. Another name of decision stump is 1-rules. Each node in decision stump represents a feature in an instance to be forecasted, and each branch represents a value that node can take. Instances are classified starting at the root node and sorting them based on their feature values.

Depending on the type of the input feature, several variations are possible. For nominal features, one may build a stump which contains a leaf for each possible feature value or a stump with the two leaves, one of which corresponds to some chosen category, and the other leaf to all the other categories. For binary features these two schemes are identical. A missing value may be treated as a yet another category.

For continuous features, usually, some threshold feature value is selected, and the stump contains two leaves — for values below and above the threshold. However, rarely, multiple thresholds may be chosen and the stump
therefore contains three or more leaves. Decision stumps are often used as components in machine learning ensemble techniques such as bagging and boosting.

2.5.4.2 C4.5

C4.5 is depth-first construction model of the decision tree with Divide and Conquer (DC) technique. It does allow performance of runtime sacrificing for the consideration of the limited memory at run time. In this C4.5 algorithm, each node in a tree is related with a set of assigned weights to take into account of unknown characteristic values. It can deal both continuous and discrete range types of attribute values. If the data is continuous, it must be discrete first. C4.5 also handles the missing values as i) Filling of missing attribute value with most common occurring value mode if its type nominal and mean if its type is numerical. ii) Assigned to a special label M for those missing nominal attributes and treat M as if it is another attribute value. This often performed poorly as compared with filling up with mean or mode.

Algorithm:

1) Check for base cases
2) For each attribute a-> find the normalized information gain from splitting on a.
3) Let a_best be the attribute with the highest normalized information gain.
4) Create a decision node that splits on a_best.

Recursive on the sub lists obtained by splitting on a_best, and add those nodes as children of node.
2.5.4.3 M5P

M5tree (Chengjun zhan, Albert Gan 2011) is the regression trees that are constructed by CART have values at their leaves, whereas trees that are built by M5 can have multivariate linear models. There are three major steps for M5 tree development i) Construction of tree ii) tree pruning and iii) tree smoothing. The M5 tree construction process attempts to maximize a measure called the standard deviation reduction (SDR). After the initial M5 tree has been constructed, tree pruning is constructed to eliminate or merge unwanted subtrees to overcome the data overfitting issue that was raised during the tree construction. During the tree construction, the M5 tree algorithm computes a linear regression model for each interior node of the unpruned tree using standard regression and by applying the attributes in the subtree. During the pruning process, attributes are dropped one by one to minimize the estimated error until the root node is reached.

A smoothing process is utilized after the tree pruning in the M5 tree algorithm to compensate for the sharp discontinuities that will inevitably occur between that inevitably occur between adjacent linear models at the leaves of the pruned tree, particularly for some models that are constructed from a small number of training cases. The smoothing first uses the leaf node to compute the predicted value then filters this value along the path back to the root node, smoothing the value at each node by combining it with the value that is predicted by the linear model. Modified M5 tree algorithm is used to handle enumerated attributes, attributes missing values, and called the new tree algorithm the M5P algorithm. M5P tree algorithm is ability to deal with categorical and continuous variables and variables with missing values. This algorithm can generate multiple linear regression models at the tree levels.
2.5.4.4 REPTree

Reduce Error Pruning Tree is a fast pruning decision tree technique and it is also based on C4.5 technique. The REPTree is used to correct the effects of noisy data on the decision tree. It is accomplished by using variance reductions to derive balanced tree splits and minimize error corrections. It can produce classification or regression trees.

2.5.4.5 CART

Classification and Regression Tree is a decision tree method that partitions a set of samples into groups and this also offers benefit for handling missing values. Its analysis has a specified outcome variable and is based on the sense of reducing impurity to build tree structure. CART is alike to C4.5 techniques but Gini index is used as divide criteria.

Algorithm:

i) Rules are selected based on how well splits based on variables’ values can differentiate observations based on the dependent variable

ii) Once a rule is selected and splits a node into two, the same logic is applied to each “child” node

iii) Splitting stops when CART detects no further gain can be made, or some pre-set stopping rules are met

2.5.4.6 K*

K* (John. G. Cleary, Leonard E, Trigg, 1995) is one of the nearest neighbour methods is shown in Figure 2.5. This works by assigning to
unclassified sample point with the classification point and classification of the nearest of a set previously classified points. Nearest neighbour methods are sometimes referred to as memory-based reasoning or instance based learning or CASE based learning techniques and have been used for classification tasks.

Instance based learning have three characteristics; a similarity function (how close together the two instances are), a typical instance selection function (which instances to keep as examples), and a classification function (deciding how a new case relates to the learned cases). The entire training set is stored in memory. To classify a new instance, the Euclidean distance is compared between the instance and each stored training instance and the new instance is assigned the class of the nearest neighbouring instance. More generally, the k* are computed, and the new instance is assigned the class that is the most frequent among the k nearest neighbours.

To classify an unknown pattern, the K* approach looks at a collection of the k nearest points and use the voting mechanism to select between them, instead of looking at the single nearest point and classifying according to that with ties broken at random. If there are ties for the kth nearest observations, all candidates are included in the vote.

$$K^*(b/a) = -\log_2 P^*(b/a)$$  \hfill (2.13)

Let I be a set of instances and T a finite set of transformation on I. Each t ∈ T maps the instances: t: I→I. T contains a distinguished member σ (stop symbol) which for completeness maps instances to themselves σ(a)=a. the probability function P* is defined as the probability of all points from instance a to instance b.
2.5.4.7 Decision tree ant colony optimization model

The ant colony optimization technique has emerged recently as a novel meta-heuristic belongs to the class of problem-solving strategies derived from natural. The ant system optimization algorithm is basically a multi-agent system where low level interactions between single agents result in a complex behavior of the whole ant colony.

Feature selection (FS) is the technique of selecting a subset of relevant features for building learning models. FS provides better understanding of the data by selecting important features within the data. However, except for datasets with only a very small set of features. In this study, a FS is proposed which combines Ant Colony Optimization (ACO) with C4.5 decision tree builder. An ACO is setup for a given dataset and each ant probability selects features on the basis of pheromone and heuristic values associated with each link. When an ant completes its tour then for evaluating fitness of the sub-set of features selected by it, we use C4.5 algorithm for constructing a rule set based only on the features in the sub-set and the evaluate the accuracy of the rule set which is considered the fitness of the solution found by the ant.

In DTACO each ant chooses the appropriate attribute for splitting in each node of the constructed decision tree according to the heuristic.
function and pheromone values. The heuristic function is based on the entropy criterion, which helps ants divide the objects into two groups, connected with the analyzed attribute values. In this way, the attribute, which well separate the objects is treated as the best condition for the analyzed node. The best splitting is observed when we classified the same number of objects in the left and right sub trees with the maximum homogeneity in the decision classes. Pheromone values represent the best way (connection) from the superior to the subordinate nodes – all possible combinations in the analyzed sub trees. For each node we calculate the following values according to the objects classified using the entropy criterion of the superior node.

2.5.5 Dynamics Based Estimation

Dynamics-based techniques (Forrester 1961) are not static and they are difficult to adjust and regulate. Certain factors like project cut off date, recruiting levels, training, budgeting and design requirements all change over a period of time. This will undoubtedly affect the productivity. The model built forecast changes in project cost, manpower needs and schedules including the reusability of software if the project details are available to the software effort estimator (Abdel-Hamid and Madnick 1991).

Dynamics technique is based on simulation modeling; the outcomes are displayed as graphs. The popular model is Madachy’s system dynamics model based upon Brook’s law (Madachy 1999). This Model was represented as networks which are dynamically changing levels with positive and negative feedback loops. The flow of information is subject to change with time. Basic assumptions bin Madachy’s system dynamics model are new people involved in project are to be trained to increase productivity and experienced people are more productive than others.
2.6 CHALLENGES FACED IN COCOMO MODEL AND OTHER ESTIMATION MODELS

A variety of effort estimation models have been developed in the past few decades. This significance induces the researchers to enhance it further. Estimation predicts future it looks ahead off. This looking ahead includes uncertainty and risks because of insufficiency of data. It cannot handle exception conditions like abrupt change in the man power, design and other major parameters.

The estimations made by statistical methods produce poor results since it cannot cope up with lack of reasoning and missing information. COCOMO models seem to be promising with accurate results this to can’t afford distorted and uncertain inputs. COCOMO model ignores majority of parameters like background knowledge of partakers. Amount time spent on each phase of development is not taken into consideration. It is not possible to rate the cost drivers inaccurate rating results in inaccurate estimation. In all these models quantification of some experience and factors are very complex. It neglects hardware issues and personnel turnover levels. It is also hard to find precisely about the KDSI estimate in the early development phase.

2.7 SUMMARY

Viable approaches promise the estimating community with a novel estimation schema. Expert estimation is a frequently used method. But project overruns and break of financial plan happen frequently. It is mainly dependent on estimation survey. Analogous model does estimation from the historical database select the related completed projects; this similarity is used for obtaining the estimate for the future task. This estimation is purely dependent on the actual project characteristic and the past experience.
In general the model based estimation techniques are as good as for the purpose of budgeting, tradeoff analysis, scheduling, forecasting and organizing, and investment analysis. Decision Tree model are also used for estimation. A tree structure is constructed with which descending the tree along its appropriate path estimates the cost. Finding the relevant attributes and choosing appropriate path lead to accuracy of the estimate. ANN models learn with the given training vector pairs. The system trains the participating neurons in such a way that the network learns with historic inputs. The change in network choices like initial weight assumption, number of hidden layers and the terminating criteria are of major concern. This is also further tougher against noise and data outliers. This research overcomes such tougher issues it involves two phases in the estimation. By integrating the graphical with genetic approach improves the efficacy and performance of the effort estimation model. This chapter is concluded with an extract of literature survey.