CHAPTER -1
SOFTWARE ESTIMATION: AN INTRODUCTION

1.1 Introduction

Engineering software’s is the well defined process which lays the roadmap for development of software’s within the specified schedule and effort and with the preferred quality. It entails utmost level of analyses, hard work and the management of the two. Big scale software planning and control incorporates the usage of quantitative programming evaluation and estimation models that are situated in the lead of theory and collected historical project data. With the perpetually expanding size and unpredictability of software’s; software development has turned out to be a more rowdy process and hence desired to take care of even the simplest action in the course of building software. This process launches through assessing the approximate size, obligatory effort as well as required time for the product development and stops with the product and additional work products constructed in different phases of development. Due to the all-encompassing characteristics of software, communities of software professionals have constantly articulated their worries over their incapability to precisely forecast the cost, quality as well as schedule of a software product in progress. The troubles being appeared in the software developments is cost overrun, schedule overrun and quality deprivation. Wrong estimation indubitably results a failure in the development process. Efficient estimation is essential and important in favor of appropriate project planning as well as for process control and is one of the basic and key testing assignments in the course of developing good software. Underestimate a project may lead to quality deprivation, resource over exploitation and setting a little schedule and thus the consequences in missed deadlines. Over estimate is yet worse than the earlier condition; assign more resources to the project and hence increasing the cost of the project devoid of any scope. Therefore, a standout amongst the hugest points of the software building expertise has been to create useful and down to earth models that
productively clarify the software progression life-cycle and exactly expect the effort required and timetable of creating a software venture. Out of different cost estimation techniques the best techniques used for estimation are model based. The technique employs different parameters for estimating. These parameters required to be stratified with the organization's past projects for accurate estimation. In the former four decades, many cost estimation models came into the existence that figure software cost in terms of effort and timetable (duration) at an untimely stage of the SLC (software life cycle), i.e. Just about quickly subsequent to the prerequisites have been made. Prepared guesstimates are utilized to handle the software advancement process in favor of conveying the finished software item promptly, inside plan with craved standard. Approximately the entire of the established parametric models have been exactly receptive to genuine information from achieved software projects. These models might be ordered as particularly for custom software and other for moderately new web based software. The customary effort estimation routines evaluate the software development effort that includes advancement of software in a programming dialect, which at last interface through information documents or databases. For the most part, these sorts of software have a dynamic execution string that gives framework administrations. Interestingly, the web systems use distinctive measurements and they are centered on evaluating the required software effort belongs to event-oriented products. These web based products for the most part includes code in a high level programming dialect, symbolism, look-and-feel, data structure, route and sight and sound objects. Disregarding the excess of these models, most software advancement undertakings experiences extraordinary schedule and cost invades.

1.2 Software Cost Estimation Practice

A software cost estimation (SCE) practice can be defined as the set of procedures and techniques that is exploited by the IT or software development company’s in the direction of reaching at a reliable software cost estimate. By and large there is a set of inputs to this practice (e.g., Framework prerequisites) and a yield of effort, manpower
required and/or calendar (duration). It is extremely hard to look at the software cost estimation practice without the general context of the software advancement process being used in a given software organization. The set of systems, strategies, and measures that an association utilizes for sorting out, overseeing, and controlling software projects is called the software process. Associations have distinctive software processes, contingent upon the kind of software they are creating. For some associations, the advancement process is extremely casual; in different cases it is overall recorded and stringently checked.

A majority of models that estimates cost see this transform as being a capacity processed from a set of cost drivers (factors). These drivers are thought to be the qualities of a framework that focus the last cost of creation. In a large portion of the pushed CE systems, the essential cost driver is expected to designate the software prerequisites. Figure 1.1 depicts that, the prerequisites (requirements) are the essential info to the procedure furthermore form the foundation for the assessment. The estimate is then balanced as indicated by a number of other cost drivers, (for example, knowledge of the staff and the unpredictability of the framework) to reach at the last prophecy.

![Diagram of the SCE process](image)

Figure 1.1 standard sight of the SCE process [Vigder and Kark, (1994)]
In reality, the cost estimation procedure might be a great deal more unpredictable than that depicted in Figure 1.1. There is an interdependency between numerous things of data, all of which are applicable to the cost estimation procedure. A large number of the information points that are inputs to the cost estimation procedure are altered and yield by the procedure (figure 1.2). Accordingly, as opposed to survey the cost estimation handle as a capacity of the prerequisites, it is frequently more faultless to view this handle as attempting to fulfill a set of constraints. The inputs to the framework are a set of constraints on the prerequisites, software building design, monetary issues and so on, while the yields are a cost assessment and a set of suppositions that fulfill all the demands. This perspective permits the constraints to be forced on any of the components that influence the cost. These components run a long ways past necessities to incorporate issues, for example, delivery date, funds and software process.

![Figure 1.2: Actual sight of SCE Process][Vigder and Kark, (1994)]
The ability to create software on time is based upon a mixed bag of factors that impact a development, advancement association's capacity to convey software product in a convenient also practical style. Different factors incorporate such things as the software product forms that will be utilized, the aptitude levels of the staff (including client work force) who will be included, the computerization that will be used, also the impacts of the corporeal environment and scenario of the software business. Indeed, various components impact our capability to opportune deliver programming product with high caliber. This data could be utilized to foresee and investigate “imagine” a scenario on future projects.

1.3 Project Planning and Cost Estimation

Cost estimation is a paramount instrument that can influence the budgeting and planning of a software task. In view of the fact that, for a software project consists of a limited number of assets, the ultimate aim is to incorporate the greater part of the peculiarity of a requirements document in the ended product. A cost estimate prepared on the verge of an undertaking will assist figure out which peculiarities could be incorporated inside the resource demands of the task (e.g., Time). The prioritization of Necessities might guarantee that the largest part imperative gimmicks are incorporated in the final software. The threat to a task is decreased once the most imperative gimmicks are incorporated at the starting on the grounds that the unpredictability of an undertaking builds with its size, which implies there is a more open door for slip-ups as advancement progresses. Along these lines, software cost estimation can have an enormous effect on the project’s schedule and life cycle.

At the same time Cost estimation can have a vital impact on the asset allotment (resource allocation). It may be reasonable for an organization to assign better assets, for example, more accomplished human resource, for exorbitant projects. In order to measure the quantity of engineering and managing staff distributed to a software project in a given measure of time, the term Manpower loading is used. The greater part of time, the more
awful situation for an organization is if a costly project comes up shorter than if a less exorbitant task fizzes. At the point when estimation tools are utilized, management and development can even explore different avenues regarding exchanging off a few assets (or variables) with others while keeping the project cost consistent. Case in point, one tradeoff may be to put resources in all the more compelling integrated development environment (IDE) so the quantity of work force dealing with a project could be diminished. There is a vast effect of Cost estimation on project, management and project planning.

1.4 History

The Software cost estimation issues are considered by the software development communities subsequent to in any event the 1960s. The majority of the exploration has concentrated towards the development of formal SEE (Software Effort Estimation) models. The cost models near the beginning were normally focused around regression analysis otherwise scientifically inferred from speculations from different areas. From that point forward a good number of model building methodologies have been assessed, for example, methodologies established on Bayesian detail, neural systems, case-based reasoning, lexical investigation of requirement specifications, linear programming, profitable generation models, fuzzy logic, and blends of two or a greater amount of these models. The parametric estimation models like SLIM and COCOMO found to be the most fundamental estimation strategies at the moment. They both hold their foundation in estimation exploration led in the 1970s and 1980s and are from that point forward upgraded with new adjustment data, with the last significant release being COCOMO II in the year 2000. The estimation methodologies focused around functionality based size measures, such as, function Point, is likewise focused around exploration directed around the decade of seventy, and however is re-balanced through change size measures and diverse counting methodologies. Table 1.1 chronologically illustrates some of the benchmarks and most paramount models in the field of SCE, which are detailed in later chapters.
Table 1.1: History of the software cost estimation industry [Boehm and Valerdi, (2008)]

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Period</th>
<th>SCE Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1965</td>
<td>SDC (Systems Development Corp.)</td>
</tr>
<tr>
<td>2</td>
<td>1976</td>
<td>Putnam SLIM (Software Lifecycle Management)</td>
</tr>
<tr>
<td>3</td>
<td>1977</td>
<td>Boeing Black</td>
</tr>
<tr>
<td>4</td>
<td>1977</td>
<td>Doty</td>
</tr>
<tr>
<td>5</td>
<td>1977</td>
<td>IBM-FSD (IBM Federal Systems Division)</td>
</tr>
<tr>
<td>6</td>
<td>1977</td>
<td>RCA Price S</td>
</tr>
<tr>
<td>7</td>
<td>1977</td>
<td>Walston-Felix</td>
</tr>
<tr>
<td>8</td>
<td>1979</td>
<td>IBM function points</td>
</tr>
<tr>
<td>9</td>
<td>1981</td>
<td>Bailey-Basili Meta-Model</td>
</tr>
<tr>
<td>10</td>
<td>1981</td>
<td>COCOMO (Constructive Cost Model)</td>
</tr>
<tr>
<td>11</td>
<td>1981</td>
<td>SoftCost-R</td>
</tr>
<tr>
<td>12</td>
<td>1983</td>
<td>ESTIMACS</td>
</tr>
<tr>
<td>13</td>
<td>1983</td>
<td>Jensen/SEER (Software Evaluation and Estimation of Resources)</td>
</tr>
<tr>
<td>14</td>
<td>1985</td>
<td>SPQR (Software Productivity Quality and Reliability) Checkpoint</td>
</tr>
<tr>
<td>15</td>
<td>1997</td>
<td>COCOMO II (Post Architectural Model)</td>
</tr>
<tr>
<td>16</td>
<td>Until Today</td>
<td>A lot of new commercial tool have been released</td>
</tr>
</tbody>
</table>

1.5 Software Costing vs Development Cost

Software project cost estimation and project planning are typically completed together. The expenses of project development are are principally the expenses of the effort included, so the effort processing is utilized as a part of both the cost and the schedule estimate. Nonetheless, one might need to do some cost estimation before nitty gritty calendars are drawn up. These beginning evaluations may be utilized to create a financial plan for the task or to set a cost for the product for a client. There are
three parameters included in processing the aggregate expense of a software development project:

1) Hardware as well as software programming expenses including upkeep

2) Training and Travel expenses (cost)

3) Effort costs (the expenses of paying software professionals/engineers)

For a majority of software tasks, the overwhelming expense is the effort cost. Machines that are influential enough for program development and execution are moderately shabby. Albeit far reaching travel expenses may be required when an undertaking is developed at diverse locales, the travel expenses are typically a little part of the effort costs. Moreover, utilizing electronic interchanges frameworks, for example, email, imparted sites and videoconferencing can fundamentally diminish the travel needed. Electronic conferencing likewise implies that voyaging time is diminished and time can be utilized all the more profitably as a part of software advancement. In a project, making each other meeting utilizing a videoconferencing instead of a face to face gathering can decrease travel expenses and time by very nearly half.

The cost of required effort are not simply the compensations of the product engineers who are included in the software project. Associations figure effort costs as far as overhead expenses where they consider the aggregate expense of running the company and divide this by the number of gainful staff. Hence, the accompanying expenses are all piece of the aggregate effort cost [Sommerville I., (2004)].

Expenses of giving, air conditioning and lighting office space.

Expenses of helping staff, for example, administrators, system managers, cleaners and experts.

Expenses of communications and networking along with the expenses of conveniences, for example, a library or recreational room.
Expenses of social security and benefits of employee for example, annuities and well-being protection.

This overhead variable is as a rule at any rate double the product engineer's pay, depending on the extent of the development firm and its related overheads.

**Table 1.2 : Software pricing concerns [Sommerville I., (2004)]**

<table>
<thead>
<tr>
<th>S.No</th>
<th>Key concerns</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Uncertainty in Cost Estimate</td>
<td>In the event that an association is doubtful about its cost estimate, it might build its cost by some possibility far beyond its ordinary benefit.</td>
</tr>
<tr>
<td>2</td>
<td>Contractual terms</td>
<td>A client may be ready to permit the software professional to hold responsibility for source code and its reusability in different project activities. The value charged may then be short of what if the product source code is given over to the client.</td>
</tr>
<tr>
<td>3</td>
<td>Volatile Requirements</td>
<td>On the off chance that the changes in the requirement are predisposed, an association can bring down its cost to co-achieve an agreement. Subsequent to the granting of agreement, the charges may be high on the basis of changes to the requirements.</td>
</tr>
<tr>
<td>4</td>
<td>Fiscal Strength</td>
<td>Developers in budgetary trouble may bring down their cost to increase an agreement. It is ideal to make a littler than ordinary benefit or earn back the original investment than to go bankrupt.</td>
</tr>
<tr>
<td>5</td>
<td>Market Prospect</td>
<td>A software company may cite a low cost in light of the fact that it plans to shift into another portion of the product market. Tolerating a low benefit on one</td>
</tr>
</tbody>
</table>
venture may perhaps provide the company the prospect to make a more noteworthy benefit afterward. The understanding picked up could additionally help it create new applications.

When an application project is underway, The project incharge/manager ought to routinely overhaul their cost as well as calendar appraisals. This assists with the planning methodology and the successful utilization of assets. In the event that real consumption is essentially more noteworthy than the appraisals, then the project manager must make some move. This may include requesting extra assets/resources for the task or altering the work to be carried out.

Programming costing ought to be done impartially with the point of precisely anticipating the expense of creating the product. On the off chance that the project cost has been processed as some piece of a project offer to a client, a decision then must be made about the cost cited to the client. Traditionally, cost is basically taken a toll in addition to benefit. Nonetheless, the connection amid the cost of project and the cost to the client is not normally so straightforward.

Programming estimating must consider more extensive hierarchical, financial, political also business contemplations, for example, those indicated in table 1.2. Thusly, there may not be a straightforward relationship between the cost to the client for the programming and the advancement costs. Due to the authoritative contemplations included, project pricing ought to include senior administration (i.e., the individuals who can settle on key decisions), and additionally project managers.

1.6 The Estimator

The individual who perform the cost assessments of the project might be either straightforwardly in charge of the carrying out for a software, for instance, a manager or a
consultant. Somebody who has learning of the company and past undertakings could utilize a similarity based methodology to contrast the current task and past project, which is a typical technique for estimation for little associations and small projects. Historical information pertaining to the projects is frequently restricted to the recollection of the estimator.

Some individuals and organisations trust it is better if the appraisals are carried out by outcasts so that there is less risk of predisposition. The reality of the matter is that individuals outside an organization will probably need to manage less organization issues than individuals inside the organization. Case in point, the software engineer for an organization may need to satisfy the manager thus present a guesstimate i.e. excessively hopeful. Having an outside assessment, may be frail due to the reason that individual would have less information of the development environment. An Empirical methods for estimation would then be needed. There may be some imperviousness to utilizing an empirical methods for estimation on the grounds that there may be a few inquiry on suitability of these methods over the expert one. Finding accurate estimators are uncommon therefore it is best to get the estimates of a few individuals or tools [Lee et al., (2002)].

1.7 Functioning of Cost-Estimating Tools

There exists numerous sorts of robotized tools that knowledgeable and practiced IT professionals can utilize to create various types of guesstimates for programming projects. Case in point, an accomplished senior executive person of any software firm can make cost estimate as well as schedule plan utilizing a few of various instruments specifically, Spreadsheets, Project management (PM) tools, and Software cost estimating tools. An oftentimes made inquiry by software professionals for the sellers of such tools is " Why should buy these tools, when we have spreadsheets and other tools for project management?". The software estimating (SE) tools are separated
from all different sorts of software project management tools and broadly useful tools, for example, spreadsheets, in these key properties:

SE tools holds learning bases of large number of Software projects.
Size forecasts can be performed by SE tools, which universally useful tools cannot.
SE tools can naturally conform assessments focused around tools, dialects, and courses of action.
SE tools can foresee quality and dependability, which broadly useful tools can't.
SE tools can foresee upkeep and support costs after deployment.
SE tools can foresee (and forestall) issues much sooner than the issues really transpire.

Not at all like different sorts of PM tools, the SCE instruments which is exploited by the development firms don't rely on the mastery of the project manager or customer, albeit the produced assesements are refined by the skilled managers. The commercially used SCE tools include the collected knowledge of numerous programming activities.

Due to the connected information bases connected with commercial SE tools, tenderfoot managers or manager confronted with new sorts of projects can depict the sort of project being managed, plus the SCE tool will build an assessment focused around comparative activities got from data contained in its related learning base.

**Figure 1.3** shows the fundamental standards of present day SE tools.
The process of software estimation (SE) begins with the reckoning of the span of the project regarding either logical source code articulations, physical lines of code, function points or, now and again, each of the three measurements. The size of the project can be gotten from the assessing tool estimating rationale, supplied by clients as an unequivocal input, or got from similarity with comparable tasks put away in the learning base of the SE tool. Actually, projects that utilized iterative development or Agile paradigm, surmised size data can be made.

After the approximation of software project size, the evaluation can be created focused around the particular traits of the project being referred to. The twelve major cases of the attributes that can influence the result of the evaluation incorporate the accompanying:

1) The rate of changing the requirements of the project.

2) The familiarity and knowledge of the developers employed with similar sort of undertaking.

3) The procedure or techniques used to create the task extending from traditional waterfall to newer paradigms like Agile etc.
4) The particular exercises that will be carried out amid advancement.

5) The count of iterations, or "sprints" that will be utilized.

6) The use of particular technology or programming language.

7) The vicinity or nonappearance of reusable relics.

8) The suite of development tool utilized for the project building.

9) Work place ergonomics.

10) The geographic division of the group crosswise over numerous areas.

11) The timetable weight put on the group by customers or officials.

12) Contractual commitments regarding expenses, dates, absconds or gimmics.

the project attributes or characteristics listed above can either be provided by the client or can be derived from the comparative tasks that are stored in the learning base of the SE tools. One might say SE tools impart a percentage of the qualities of the object oriented standard in that they permit legacy of imparted ascribes project wise.

The imparted traits or attributes discussed above are called templates as far as SEE is concerned. These traits can be inherent in various ways. Case in point, a user of estimating tool can indicate a current finished project and select any or the majority of the gimmicks of that project as the basis of the template. Therefore, if the project chose as the basis of a template were a web based application, utilized the ASP.Net programming technology, and used formal plan and code investigations, these properties could be inherited as a feature of the web development cycle and could get to be a piece of an estimating template for different activities.

Templates can likewise be gotten from sets of projects instead of from one particular project, or can even be custom-constructed by the user of the estimating tool, utilizing artificial elements. On the other hand, the most widely recognized way for the development of template is to utilize the automatic template construction capacity of the
estimating tool, and essentially select significant chronicled projects to be utilized as the premise for the template.

**Figure 1.4 : Key Estimation Characteristics [Jones, (2007)]**

Generally, the templates in software estimation are concerned with four key sorts of inherited characteristics as illustrated in **figure 1.4**.

Three out of the four factors depicted in above figure are genuinely evident in their effect. But the fourth factor "Environment" is not obvious, however is similarly critical. This factor covers singular office space and the communication channels among geologically scattered development groups. Shockingly, get to a calm, clamor free office environment is one of the main considerations that impacts the Software quality and gainfulness. The capacity to incorporate ergonomic components in an assessment is an astounding sample of the value of modern software cost estimating tools. Not just do they contain the aftereffects of hundreds or a great many finished projects, however the tools
contain information about powerful elements that numerous project managers may not completely get.

1.8 General Structure of Estimation Models

A classic estimation model is inferred utilizing regression examination on information gathered from past software projects. The general formation of such models takes the structure [Pressman, (2010)].

\[
E = A + B \times (\text{something})
\]

where A, B, and C are experimentally inferred constants, E represents required effort in P-M (person-month), and ESTv represent the estimation variable. Notwithstanding the relationship noted in above equation, the greater part of estimation models have some manifestation of project adjustment components that empowers E to be balanced by other task characteristics. Table 1.3 highlights the list of some well known and referred estimation models among the numerous LOC and FP oriented estimation models proposed in the literature.

Table 1.3: Famous Estimation Model Equation [Pressman, (2010)]

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Model Proposed by</th>
<th>Model Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Walston-Felix model</td>
<td>( E = 5.2 \times (KLOC)^{0.91} )</td>
</tr>
<tr>
<td>2</td>
<td>Bailey-Basili model</td>
<td>( E = 5.5 + 0.73 \times (KLOC)^{1.16} )</td>
</tr>
<tr>
<td>3</td>
<td>Boehm simple model</td>
<td>( E = 3.2 \times (KLOC)^{1.05} )</td>
</tr>
<tr>
<td>4</td>
<td>Albrecht and Gaffney model</td>
<td>( E = -91.4 + 0.355 \times FP )</td>
</tr>
<tr>
<td>5</td>
<td>Kemerer model</td>
<td>( E = -37 + 0.96 \times FP )</td>
</tr>
</tbody>
</table>
1.9 Sequence of Estimation

According to Caper Jones [Jones, (2007)], there is a standard succession for software cost estimation. This succession can be utilized with manual software cost estimates, furthermore reflects the estimation organizes in the software-estimation tools. Figure 1.5 illustrates the ten steps of the estimation sequence and the task execution of each step is briefly described underneath.

Step - 1

To perform Software cost estimation for the project, the thorough requirement analysis is primarily required. Along these lines, before assessing itself can start, it is important to investigate and comprehend the client prerequisites. Eventually it ought to be conceivable to make estimates consequently from the prerequisites details, yet the current level of estimating tools requests human intercession.

A typical estimating action today is to investigate the software requirements and make the total of function points focused around those prerequisites. This gives the essential size information utilized for formal cost estimation. Generally certified function point counting professionals are employed to carry out this task.

Step – 2

Each manifestation of estimation and each software cost-evaluating tool needs the sizes of key deliverable s with a specific end goal to finish an estimate. Size information can be inferred in a few ways, including the followings:

An estimating tools inherent sizing calculations.

Similarity with comparable projects of known size.

Utilizing "Project manager's instinct".

Utilizing "developer's instinct".

Extrapolation from the totality of function point.
Utilizing statistical methods or Monte Carlo simulation.

1: Analyze the Requirements
2: Start with Sizing
3: Identify the Activities to Be Included
4: Estimate Software Defect Potentials and Removal Methods
5: Estimate Staffing Requirements
6: Adjust Assumptions Based on Capabilities and Experience
7: Estimate Development Costs
8: Estimate Maintenance and Enhancement Costs
9: Present Your Estimate to the Client and Defend It Against Rejection
10: Estimate Effort and Schedules

Figure 1.5: Cost Estimation Sequence

Source code estimating must be custom-made to particular programming technology, and more than 6500 languages are presently being used and they are growing with the rate of around 20 per year. Around 33 percent of software projects use more than a solitary programming language.
Sizing is a key activity in cost estimation. On the off chance that the sizes of significant deliverables can be anticipated inside 5% to 10% accuracy, then the exactness of the overall estimate can be great. In the event that size forecasts are fiercely off base, then whatever is left of the estimation will be wrong, as well. Empirical proof from expansive software projects demonstrates that prerequisites develop at a normal rate of around 2 percent for every timetable month from the end of the requirements phase until the begin of the testing stage. The aggregate development of prerequisites can surpass 50% of the volume of the early requirements when measured with function points. A real issue with estimation exactness has been disregarding or forgetting requirements creep. Cutting edge cost-assessing instruments can anticipate requirements growth and can incorporate their costs and calendars in the appraisal.

**Step – 3**

When the sizes of key deliverables are identified, the following step is to choose the set of activities that are going to be performed. The term activities alludes to the work that will be performed for the project being evaluated, for example, requirements, internal design, external design, design inspections, coding, code inspections, user document creation etc. Exact estimation is incomprehensible without information of the activities that are going to be used.

The patterns of the activity differ generally from project to project. Expansive frameworks use a lot greater number of activities than do little projects. The projects developed using waterfall method of software development use a bigger number of activities than the projects developed via Agile methods.

**Step – 4**

The most extravagant and tedious work of software advancement is the work of discovering bugs and altering them. Normal defect removal efficiency before delivery of
the finished product is 85%. The cost of discovering and repairing these imperfections averages around 35% of the aggregate development cost of developing the application. Deformity repair costs and schedules are regularly bigger than coding costs and calendars. Exactness in software cost estimates is impractical if defects and defect removal are excluded in the estimates.

Defect estimation incorporates prescient capacities for prerequisites defects, design defects, coding deformities, user documentation deformities and an extremely upsetting class called bad fix deformity. The word "Bad fix" alludes to new deserts coincidentally infused as an issue result of repairing past deformities. Bad fix deformity normal around 7 percent of all deformity repairs. few estimating tools can foresee bad fixes.

The capacity to anticipate software imperfections would not be exceptionally helpful without an alternate sort of estimation, which is anticipating the deformity evacuation productivity of different sorts of audits, reviews, and test stages. Cutting edge software cost-evaluating tools can anticipate what number of bugs will be found by every manifestation of deformity evacuation, from work area checking through external beta testing.

**Step – 5**

Each software deliverable has a trademark task degree, or measure of work that is possible by a solitary worker. Case in point, a normal task for an individual developer will extend from 50000 to 15,000 source code proclamations. Then again, vast frameworks likewise use numerous expertise or specialists, for example, framework architect, database admin, quality assurance specialists, software engineers, technical writers, testers, and so forth. Distinguishing every classification of skilled laborer and the quantities of specialists for the overall project is the subsequent step in software cost estimation.

Staffing necessities rely on the exercises that will be performed and the deliverables that will be made, so staffing expectations are gotten from learning of the overall size of the
application and the action sets that will be incorporated. Staffing expectations likewise need to be mindful of "pair programming", which are part of a few of the recent Agile methods.

For extensive frameworks, developers themselves may involve short of what a large portion of the workforce. Different sorts of specialists and project managers include the other half. In the event that the task is sufficiently huge to need experts, precise estimation obliges that their exertions be incorporated. Equally programming and different sorts of noncoding exercises, for example, creation of manuals and quality certification, must be incorporated to finish the estimate effectively.

**Step – 6**

There exists a wide range of Software professionals beginning with top specialists with a vast experience to entry level graduates doing their first assignment. Once the classifications of specialized professionals have been recognized, the subsequent step is to make changes focused around the particular experience levels and skill set. Specialists can tackle more work, and perform it quicker, than can beginners. This implies that specialists will have bigger task degrees and higher generation rates than normal or unpracticed staff. Different conformities incorporate work hours for every day, leaves and breaks, and suspicions about the geographic dispersal of the software group. Changing the estimation to match the abilities of the group is one of the more discriminating estimating activities.

Even as estimating tools can make conformity to match differing degrees of aptitude, these tools have no chance to get of knowing the particular abilities of any given group. Numerous business estimating tools default to "normal" abilities, and permit clients to change this suspicion upward or descending to match particular group qualities.
Step – 7

The effort estimation and schedule estimation are nearly coupled, and are typically performed in an iterative way. Exact effort estimation obliges information of the essential size of the application in addition to the numbers and experience levels of the development team. Exact timetable estimation obliges information of the exercises that will be performed, the quantity of augmentations or "sprints" that will be done, the sizes of different deliverables, the cover between exercises with mutual dependencies, along with the exact numbers of software professionals employed in the project with their skill sets.

However the communication between Schedule and effort measurements is convoluted and now and then is nonsensical. For instance, if a web based application will take eight months in the event that it is created by one developer, including a second software engineer won't slice the calendar to four months. Undoubtedly, a point can be arrived at where putting on extra developer may ease off the project's timetable as opposed to quickening it. The critical rules and logic that connect effort and schedules for any software tasks are the core part of the software cost-assessing tools. For instance, adding a software human resource to a software project inside one division will generally abbreviate development schedules. However in the event that enough staff are included so that a second division is included, timetables will extend. The purpose behind this is that software plans, furthermore gainfulness rates, are contrarily related to the number of project managers.

Actually, for vast software ventures with numerous work groups, the rate at which productivity of development decreases has a tendency to associate all the more nearly to the number of managers that are locked in than to the genuine number of developers included.
Step – 8

Calculating development costs are the by last phase of estimation and are exceptionally intricate. Development costs are clearly dependent upon the effort and schedule for software ventures, so these elements are anticipated to begin with, and afterward costs are connected a short time later. Software project's cost that require precisely the same measure of effort as far as hours or months can shift generally because of the accompanying reasons:

Normal compensations of developers and project managers.
The overhead rate connected to the project.
Currency trade rates, if the software assignment is produced globally.
License charges for any procured software required.
accomodation and conveyance costs for new employees.
Travel costs for on site projects or projects created in distinctive geographical areas
Legal expenses for copyrights, licenses, or different matters
Marketing and promoting costs
Training cost
If the project is a web based application, than cost of Content acquisition.

In general, creating a full and perfect cost estimate for a software undertaking is significantly more mind boggling than creating a resource estimate of the quantity of work hours that are liable to be required. Numerous cost components, for example, travel, are just by implication identified with effort and can affect the last cost of the venture essentially.
**Step – 9**

The Software products regularly keep on being utilized and calibrated for a long time. Maintenance and enhancement cost estimation are exceptional themes, and are more mind boggling than fresh project cost estimation. Evaluating upkeep costs obliges information of the likely number of clients of the application, consolidated with learning of the plausible number of bugs or absconds in the software at the launching time. The assessment of enhancement costs obliges great authentic information on the rate of change of comparative projects once they enter generation and begin being utilized. Case in point, new software products can include 1% or all the more altogether volume of new peculiarities with each one release for a few releases consecutively, yet then back off for a time of two to three years prior to another major release occurs.

Numerous estimating tools can assess both the beginning development costs of an undertaking and the upkeep and improvement cost designs for more than five years of use by clients. There is no real utmost on the number of years that can be assessed, but since long-run projections of client numbers and conceivable new peculiarities are profoundly questionable.

**Step – 10**

After finalization of the cost assessment, the following step is to present the estimate to the customer who is going to fund the project. For extensive frameworks and applications of around 500,000 SLOC, around 60% of the starting estimates will be rejected by the customer. Most of the times, the customer will announce a particular delivery date much shorter than the evaluated duration. Also, the customer will declare that costs must be held inside points of confinement much lower than the assessed costs. The software projects where formal appraisals are rejected and supplanted by subjective timetables and costs got from outside business needs instead of team capabilities have the most astounding disappointment rates in the business. Around 60% of such activities will be drop and never finished whatsoever. (At the purpose of
dropping, both costs and calendars will as of now have surpassed their focuses.) The remaining 40% of projects that at long last do get finished, the normal timetable will be around 1 year late, and the normal cost will be around 50% higher than targets. The best safeguard against having a cost evaluation rejected is to have robust verifiable information from at any rate more than dozen comparable projects. An alternate way is to set up a full activity based estimate that incorporates quality, paperwork, creeping requirements, development activities, and all support cost etc.

1.10 The Estimating Life-cycle

To begin with, it is essential to perceive the constraints of software cost estimating at the macro level. As demonstrated in the figure 1.6, the distinctive exactness of cost estimates differs focused around the current software advancement stage. According to William Roetzheim [Roetzheim, (2012)], Early vulnerability in the estimate is to a great extent focused around fluctuations in the input parameters to the estimate. Later instability in the estimate is focused around the fluctuations of the assessing models. At first, at the idea stage, you may be displayed with an ambiguous meaning of the project. In spite of the fact that the requirements may not yet be completely comprehended, the general purpose of the new software can be perceived. Here, estimates with an exactness of +/- 50% average for an accomplished estimator utilizing informal strategies. After the requirements are sensibly well comprehended, a function oriented assessment may be arranged. By then, estimates with an exactness of +/- 25% are typical for an accomplished estimator, utilizing different well established procedures and techniques.
At long last, after the detailed configuration is finished, an execution oriented estimate may be created. This estimate is commonly precise inside +/- 10%. The key is to perceive the imperativeness of periodic re estimates all through the task lifecycle, accordingly recognizing issues sufficiently early to make restorative move [Roetzhem, (2012)].

11 Uncertainty in Effort Estimation

How an estimator come to know about the accuracy of his/her estimates, Whether the estimator tracks their assesements through the software project's life cycle and comprehend where he over or under estimated so he can improve in the further assignment.

[Molokken and Jørgensen, (2003)], examined results of software estimation and concluded reviews as most regularly reporting that 30 to 40 % of projects invade. They presume that the invades' underlying drivers are intricate, the information isn't generally solid, and those reacting to the studies "may tend to over report causes that lie outside of their obligation, for instance, client related reasons." It is frequently most straightforward to credit the overwhelm to factorsover which an individual have little control—that is, the
client. The 2009 Standish Group’s Chaos Report cites project invades (overruns) on 44% of projects.

Figure 1.7 illustrates that in the early phase of projects, feasibility study; a team member can undervalue the size of a project by up to 4 times its finishing size, or overvalue it by the same range.

![Figure 1.7: Uncertainty in estimation during a project life span [Sommerville, (2001)]](image)

One conceivable foundation for the event of this circumstance is due to the fact that the current cost calculation models and tools are deficient and insufficient to cede impeccable estimates. In view of the verity that extent of Software expenses is swelling above the cost of hardware [Boehm, (1981)], and the advent of relatively newer methodologies such as Agile, of software development, a huge challenge in front of software development communities has been flawlessly speculating effort and duration of software. Considering the verity that agile become all the more intriguing to be embraced as new technique in software development and project planning approach, it is noteworthy to take a gander
on making of estimation framework for such newer approaches. Light-footed techniques
give direct process as contrast with traditional development strategy. With agile, the
development methodology ends up being easier and less demanding. Most of the analysts
concurred that Agile offers straightforward steps and a considerable measure simpler
contrasted with the customary technique [Abrahamsson, (2003)], [Stephen, (2009)],
[Chandra et al., (2009), [Green et.al, (2010)].

1.12 Software success scenario

The essential issue in software development undertaking is the means by which to finish a
task in particular time, plan and assets. With a specific end goal to determine these three
characteristics are accomplished, individual who included in software estimation
transform particularly, extend administrator need to measure all necessities are viewed as
and generally characterized included undertaking timetable overwhelm is one of the
principal supporters to extend disappointed. Legitimate and complete asset identification
is required to begin estimation process. With attention to every last one of the
prerequisites, the cost estimation methodology gets to be less demanding and will handle
the faultless result [Brien, (2004)],[Green et.al, (2010)].

In as something to be shared practice, SCE procedure is vital near the beginning stage of
programming advancement. Nonetheless, it could keep going in different stages for
instance being developed and organization process. This is because of progressions of
requirements and prerequisites every now and then. The correctness of software cost
estimation are relying upon how software advancement groups characterized the assets
required and the amount of the asset. Implying that, they have to completely investigate
these two components throughout the task arranging stages. The essence of precise
software cost estimation is given below [Zhang et al., (2006)].

(i) It can help to characterize and prioritize advancement ventures as for a general
marketable strategy.
(ii) It could be utilized to figure out what assets to focus on the task along with optimum utilization of these assets.

(iii) It might be utilized to evaluate the effect of progressions and help re-arranging.

(iv) Activities might be less demanding to oversee as well as a control while assets are improvably synchronized to genuine prerequisites.

(v) Clients anticipate that genuine improvement expenses will be in accordance with assessing expenses.

A study made by Standish Group outlines that just 29% of IT anticipate with ordinary techniques were achieved on-time and on-plan with all gimmicks and functionality tagged. Notwithstanding, 53% of the ventures were done over-plan, over the assessed time, and/or giving less gimmicks and capacities; 18% of projects were lost eventually the development cycle. It is affirmed in a latest study carried out by the same group that just 32% of software ventures be closed promptly, on the plan along with all enveloping gimmicks and capacities detailed initially. Additionally, they found that 24% of the undertakings fizzled and the software projects that were challenged increased to approximately 44% [Standish Group, (2009)]. Figure 1.8 demonstrates the exploration result.
The figures point toward great problems with software engineering projects and have had a massive impact on web application software development. They recommend many efforts and best practices put onward to improve how software industries develop software are barely successful.

The literature records a few regular reasons for cost overwhelms and software failure. The essential drivers of software project overwhelms are listed underneath [Laird, (2006)].

**Lack of education and training**: Numerous individuals don't know how to estimate, have no preparation in estimation, and get no criticism on their assessments to help them move forward. An engineer who knows how to compose code well doesn't essentially know how to estimate.
Confusion of the craved effort/schedule focus with the evaluation:
Development groups are much of the time pushed into dates due to business needs instead of a sane plan to handover on schedule dates.

Hope based planning: Engineers know the "right reply": that the undertaking is on time and on plan, taking into account the management assigned target.

Software professional's failure to solidly convey and upport their estimates:
The absence of fine estimation procedures and information, every now and again prompts being pushed into the "briefest timetable one can't demonstrate one won't make" rather than a levelheaded calendar focused around likely results.

Incomplete, varying, and crawling requirements: Nothing damages a fixed-price and settled calendar extend more than changing and developing requirements.

Quality shocks: Software Projects can without much of a stretch invest a large portion of their time in the test-and-fix stage, particularly when the requirement for speedy development causes the development group to go for broke and turn over insufficiently tested code.

1.13 Need of Study

Erroneous software development effort estimates are a standout amongst the most essential reasons of IT-project disappointments. The historical backdrop of the software business is besieged by conflict stories concerning tasks which have significantly over-run their financial plan and neglected to meet the customer's desires. While excessively
low effort appraisals may prompt project administration issues, deferred deliveries, plan
invade in addition to poor software quality; excessively high effort evaluations possibly
will prompt vanished business prospects as well as wasteful utilization of assets. A few
studies have observed that around two-thirds of all activities generously overwhelm their
estimates [McConnell, (1996)]. Further, that the normal expansive undertaking misses its
release date by 25 to 50 %, and the size of the normal timetable slip expands with the
span of the software project.

These issues propelled us to direct research that goes for critical improvement in web
effort estimation strategies.

1.14 Type of study

The nature of the study carried out in this thesis is a combination of “Case study” and
“Controlled experiment”. The study encompasses a method for definite exploratory
examinations, both prospectively and reflectively, that endeavor to comprehend and
clarify questions or test speculations, utilizing principally qualitative analysis"

1.15 Problem Domain

As of 2012 there are no effective metrics for quantifying the size and effort of web
content such as images, music, photographs and animation for agile paradigm. As a
result, there are no solid economic studies on the cost of ownership of agile web
application. The focus of this thesis is to analyze the existing software cost estimation
models, explain the drawbacks of these models intended for software engineering data
and suggest the AGILEMOW, an extension of COCOMO II [Boehm et al., (2000)] and
WebMO [Reifer, (2000)] model which allays a couple of the issues confronted by agile
practitioners in web based software development.
Figure 1.9: Research Domain of Knowledge

Figure 1.9 signifies the web cost estimation within the exploration area of learning in view of the fact that the real demand is to obtain the information from scholastics that are concentrating development of web based application. The presented model is produced utilizing a combination of professional opinion in addition to real information from completed academic projects developed by computer science students of undergraduate studies. It facilitates the agile team to capture the characteristics of the web application as well as Agile methods. The work is indicating little and medium sized web based software (applications) created in general new Agile situations. This sort of ventures speaks to a respectable number of the undertakings that are, no doubt created in commercial enterprises today. Agile web engineers confront a much focused business where customers need cost appraisals, subject to brief time for their ventures, and these evaluations ought to be exact enough for the web development to be financially savvy. The significant question answered in this thesis is: “Is the AGILEMOW model for web cost estimation of Agile based projects is better than the already existing WebMO model of web cost estimation?” The answers in this thesis indicate that the prescient execution of the AGILEMOW model of our recent sample of seven completed web projects is impressively superior to that of the Reifer’s WebMO. The AGILEMOW model calibrated
utilizing seven web projects yields MMRE of only 33.28% when validated against 7 projects, whereas the WebMO model yields MMRE of 49.15% on the same data.

1.16 Objectives of the Research

The major objectives in this thesis are as follows:

A Radical analysis of existing methods that are utilized for constructing SE models with a focal point on the experimental alignment of the model.
Inspecting diverse size estimation methods and design of a new sizing metric which better represent the volume of web Agile projects, which will act as a key factor for effort estimation.
Proposing of a tailored version of the effort estimation model which holds the characteristics of an Agile environment efficiently.
Validation of the proposed model using data set of Agile projects and compares the performance with the existing one.

1.17 Significance and Scope of research

Despite the advancement in software improvement methods and software cost estimation tools, correct estimation remains a troublesome errand. A percentage of the trouble is because of the natural uncertainty and unpredictability of vast software application. Yet a portion of the trouble connected with faultless estimation is self caused and because of absence of research into key problem areas.

The Agile and web development methodologies have created intriguing and extraordinary examples of advancement activities that are not found in standard waterfall based projects. Starting now, there is a deficiency of strong experimental information on
the size, effort and duration of these moderately newer activities. The proposed research study is very significant for achieving both a better understanding of web software development along with a better understanding of estimation of web based projects developed under the agile umbrella.

1.18 Outline of the Thesis

The contribution of this thesis is organized into eight chapters, each briefly described as follows

Chapter 1: Software Cost Estimation: An Introduction

An introduction/prologue to the research area “Software cost estimation ” is given in this chapter. The terminology and importance of cost estimation is elucidated in detail. The precise steps of estimation sequence is described. Chapter also focuses on issues and challenges in the field of SCE. In the later part, objectives and significance of the research are acknowledged. Need of accurate estimates for software development is emphasized briefly and the requirement of tailored measures of effort models to deal with a new development environment is discussed.

Chapter 2: Literature Review

This chapter gives top to bottom audits of the research work in the area of software cost estimation as reported in the literature mainly in research journals and articles. Significant work of various researchers’ and their contribution towards improvements in effort estimation strategies (web cost estimation) are reported in this chapter. Much needed emphasis is given to the literature pertaining to parametric models. Gaps have been identified and finally, based on this, the problem is formulated.

Chapter 3: A Radical analysis of existing software cost estimation models and practices
This chapter covers the discussions on the existing scenario of typical software Cost estimation, the leading techniques and their comparative strength and drawbacks of the existing scenario. The chapter concentrates on the association of effectively existing benchmark strategies into five real classes, giving layout samples of every class.

Chapter 4: Agile Software development Scenario in web perspective

This chapter focuses on the principles and manifestoes of Agile paradigm and provides a concise and basic introduction to the Agile methods along with discussion about the existing estimation scenario in this new paradigm. This chapter partially forms the foundation for the design and development of “AGILEMOW” model which is described in chapter 6.

Chapter 5: Web Based Software development and Estimation conundrum

This chapter’s aim is to introduce a Web based application, web stats along with the web architecture and distinct features of Web Apps. Web development in contrast with traditional software development is discussed. An analytical review of web cost estimation is presented trailed by a discussion of Web size and web effort measures.

Chapter 6: Effort Estimation of Agile Web based projects using proposed AGILEMOW Model

This chapter talks about the inevitability for newfangled measurements and model to fast predict the size as well as effort for agile based web development projects. The proposed method in this chapter is called web estimation using COCOMOII for Agile method (AGILEMOW). It delivers a need to acquire the effort estimation in a constrained period utilizing restricted data. Unexpectedly with other accessible models, AGILEMOW utilizes light-footed qualities, crude recorded data concerning development ability and elevated granularity information regarding the web framework to be created. This research focuses on development of effort estimation model for agile web projects. Creation and use of the model is enlightened in detail. Sizing the Agile web application is defined as AWP (Agile Web Points). Further the identification of Agile specific cost
drivers are shown and their assessment is described in detail. The proposed model is calibrated using the experimental data gathered from seven completed web projects. The experimental results prove that the model has good estimation accuracy in terms of effort and MMRE. The suggested method/model is straightforward and particularly appropriate for little or medium size electronic frameworks developed by making use of agile methodologies.

**Chapter 7: Results and Discussion**
This chapter summarized the results obtained from the research followed by discussions on the results.

**Chapter 8: Summery and Conclusions**
This chapter contains the summary of the research work carried out, conclusions and recommendations on the findings made. Limitations pertaining to the work are also discussed along with future scope and extension of the work.