RESEARCH REVIEW JOURNALS
Volume-04
Issue-05
May-2019

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com[UGC Listed Journal]

A Review Paper on Software Development Estimation Models

[1]Binod Kumar Acharya and [2]Dr. Jitendra Sheetlani

[1]Research Scholar, Computer Science, Sri Satya Sai University of Technology & Medical Sciences, Sehore, M.P., Sehore
[2]Research Guide, Computer Science, Sri Satya Sai University of Technology & Medical Sciences, Sehore, M.P., Sehore

ARTICLE DETAILS

Article History
Published Online: 25 May 2019

Keywords
Software, Project Management, Effort Estimation, Software Development Effort Estimation.

ABSTRACT

In the software industry, Software effort estimation is essential. It's become increasingly difficult to estimate the cost of software development as software has increased in size and complexity. In past years, this was a dilemma. The fastest-changing aspect of software development has made it difficult to create parametric models that produce high precision for software development in all domains. The models and techniques for estimating software costs are summarized in this paper. There is no particular strategy that is best for all situations, and that a careful analysis of the results of many methods is the most likely produce realistic estimates. The use of workforce is measured in terms of effort and is described as the total time it takes for members of the development team to complete a task. Man-day, man-month, and man-year are common units. This value is significant because it is used to estimate other important aspects of software projects, such as cost and overall time to complete a project. This paper discusses a study that compared various estimation methods.

1. Introduction

In the software development process, the cost estimation of the software process needs to be accurate. The project management that the company is doing are leading to their product being delivered on time to the customer, which in turn will lead to customer satisfaction.

Most of the recent research on the software engineering profession focuses on determining the minimum effort required in building a large software system (Putnam 1978; Shepperd & Schofield 1997; Jorgensen & Shepperd 2007; Minku & Yao 2013). For this software project, the cost estimates include the manpower loading and an effort to completion, and the duration of the process (Vigder & Kark 1994). Manpower loading refers to number of skilled personnel allocated to the project as a function of time. Effort is the engineering and management effort required to complete a project, usually measured in terms of Person Months (PM). The duration of the job is how long it will take for that project to be completed. As a general rule, wider projects carry with them larger error rates. In Boehm (1988), there was suggested understanding and controlling software cost/costs was extremely important. The "black box" approach is used to uncover low-cost components in object-oriented software. Black box is otherwise called as back-prop, an influence technique, or the LAPS approach. It analyses the overall results of a number of software projects which distinguish the different factors of software costs such as the objectives the company's team wants to achieve, the experience of the company's people, and the capabilities of the software being developed. Alternative terms and shortened versions of "software project budgeting," including "project budgeting," "project cost estimation," "software project costing," "software project cost management," "software project cost estimation," "project budgeting," "project cost management," "project estimation," "software project cost budgeting," "software project cost control," or "software project cost and schedule control."

Software managers are interested in estimating the cost involved in software development, which has a significant effect on the success or failure of the project. A vast estimation error in the software costs directly affects the profit and loss of the project. Furthermore, the uncertainty of the estimation is what causes overestimation or underestimation of effort. When resources are excessively used, people are not able to meet the objectives that were set for them. Since our customers are not satisfied with the whole invoice, we should do accurate estimation given that we failed it. Summarizing and estimating may potentially delay the project completion and project manager resources, such as staff, time, and cost.

The primary goal of a software development project is to increase worker productivity. Productivity is the ratio of the amount of goods or services produced, or how much wealth/income is created, to the labor/expense that went into producing them (Jones 1996). Technical and business analysts often choose one of the many methodologies to measure productivity such as Function Point Analysis, Management By Object Model (MoM) and COCOMO. The size of the software project is proportional to the productivity, time, and effort to deliver the software. Research has shown that performance of a team is improved significantly above 10 people. Performance degrades with increases in team size up to 11. (Putnam 1978).

When looking at the feature itself, the smaller the code, the higher the KLOC. KLOC is used to count and analyse the number of lines in the source code and ranks the software on the complexity of the programme. There are two kinds of Lines of Code (LOCs), Physical LOCs and Logical LOCs. (In fact, Loci consists of both LOC types.) You may display the line number to indicate the first non-comment section or the line number where the first blank line appears in a source code. A logical logic count is used to measure the number of statements.

In 1970, Albrecht developed the Function Points metric, which assigns a value to the functionality of the software. Different kind of classes are known as an FP type class. External Inputs (EIs), External Outputs (EOs), External Inquiries (EQs), Internal Logical Files (ILFs) and External Interface Files (EIFs) are all found to be included in the different classes. Transactional function types, designated as EIs, EOs, and EQs, are denoted by their letter. And as for ILFs and EIFs, they are called data function types. (Albrecht1979).

The effort multiplier (EM) and cost driver issue is an important consideration in software estimation. However, the most of the estimation techniques and approach that were proposed neither manage to obtain successful cost forecasts, nor resolve to an explicit, measurable, and concise set of factors that are affecting productivity. Because a variety of project attributes affect the time it takes to develop software, one of the main challenges of building your software system is to understand and quantify the effect of these project attributes on the time required to develop the system. Vigder and Kark (1994) stated that regulatory agencies are integrated with their regulatory structures and their administrative processes, from the inception of the political process, public policy and their environment, to the decision-making states. Important parts of what goes into the costing process are also included in the process, like the changing numbers assigned to them, which are often outputs to the costing process.

First, the system requirements, delivery date, financial and manpower resources, system architecture, software process, and the environment in which the software will be installed. Jones (1998) states that the following major categories must be taken in consideration when estimating the cost of software development: a) The size of the software code, b) the changing software requirements, c) the number of defects, including those found in the software's testing phase, d) the capabilities of the software development team, e) the development team's planned compensation (salary and overheads), f) the tools in use and their configuration, and g) the development activities..

The various types of software cost estimation tools and methods as suggested by Jones (1998) are listed as follows:

SEER software cost estimation tool, PRICE-S Software estimation tool, IBMFP metrics, SLIM tool,COCOMO Model, DeMacro FP, Mark II FP, ESTIMACS, Backfiring, SPQR/20 (Software estimation tool), SPR feature points

2. Some existing techniques in the effort estimation

You can use estimation approaches that start with both top down and bottom up estimates, using a person's experience and their professional knowledge. You can use a model they usually handle, or use a statistical modelling approach. You can use a regression-based approach, or a statistically based approach. You can use Bayesian estimation techniques, or any other estimation technique. You can even use all these approaches at the same time. Although studies are based on expertise, learning-oriented techniques are discussed in this chapter. Developing a model is based on using regression techniques to boot. Algorithmic or Parametric models use mathematical formula to estimate the forecast.

The system wide approach is the starting point for a top down approach. Similar to this approach, a product can be analysed for the overall functionality of the product, and how that functionality is provided by the smaller sub-functions. See the integration, configuration management, and documentation of the system that was absorbed into the ongoing system, becoming part of the fundamental system (e.g., Putnam model). As the starting point of bottom up approach, the component level is important. The subject is decomposed into several components, or parts, which are made up of several smaller parts, which are then composes of even smaller parts, which are then divided into smaller parts, and so on. Effort estimation is calculated by not only considering the development of each single component of a system. Effort required for the whole system development is referred as the sum of all component costs (e.g., COCOMO's detailed model). Since changeability is an essential in dynamic modelling techniques, they recognise that their estimations at one time have much more significant factors than at another time altogether different. Maximum entropy is framed on Shannon's "dynamic model" of information, most notable amongst it are the "algorithmic stochastic dynamics" of L. Shannon. Planning and controlling operations are very good for Dynamic based estimation. The biggest disadvantage of this solution is that it is difficult to calibrate. Vigder and Kar are investigating the factors involved with inaccurate software cost estimations include problems with the requirements, issues in recruiting, establishing an effective maintenance programme, problems with budget, system size, software process and software project advancement in itself, problems with the systems maturity, the effects of the software process, the lack of negative data about the software project progress, and the lack of system application domain know-how. The factors that affect pricing and value of a software product include market opportunity, cost estimate uncertainty, contractual terms, requirements volatility, and financial health (Sommerville 2004).

Dheeraj & Gupta(2016), the method of estimating the work taken to implement a software system is known as software cost estimation. Cost estimation would remain a difficult challenge, and researchers should devote their efforts to developing new techniques for this mission. The consistency of the firm's software expenditure decisions is directly influenced by the precision of the software project cost estimation. Accurate cost analysis will help you save money and improve your company's productivity. As a result, over the last 20 years, a slew of estimation models have been proposed. This paper focuses on a brief analysis of the literature on different cost estimating methods, as well as current developments in the field. This is an area that has attracted a lot of interest from researchers in the field.

Carlos et. al.(2015), the field of software-development effort calculation investigates methods for estimating effort using prediction techniques. Despite the fact that this discipline has a significant effect on business budgeting and project preparation, the number of works categorizing and analyzing currently existing methods is still restricted. As a result, the aim of this article is to provide a concise review of these approaches, as well as to identify research holes, problems, and trends. Method: Using well-established functional

principles, a comprehensive mapping of the literature was planned and carried out. After applying a careful filtering method to a pool of 3,746 candidate studies, 120 primary studies were picked, evaluated, and classified to address six study questions. Results: Over 70% of the studies used multiple effort estimation approaches; over 45 percent used measurement research as a research method; over 90% of the researchers were students rather than professionals; the majority of the findings were judged to be of good quality, and most were presented in journals. Conclusions: Our project assists clinicians and scholars by offering a body of information about existing literature, which can be used as a springboard for future research. This post discusses issues worth looking at, such as the use of cognitive load and team engagement.

Nerkar & Yawalkar(2014) The method of calculating the work and cost needed to produce software is known as software project estimation. One of the most critical aspects of software project management is software cost estimate. This study paper gives an overview of software cost analysis approaches that use both algorithmic and non-algorithmic models. COCOMO model, Putnam Model, Function-Point dependent Model, expert decision, estimation by comparison, Parkinson's Law, and pricing to win are some of the methods used. Each measurement method has its own set of advantages and disadvantages. It also gives a description of the Hybrid Model. The key goal of this paper is to clarify all of the current automated cost estimating techniques. Really, only one strategy isn't the right for all situations when it comes to generating realistic figures.

Yansi et. al.(2014), if software products are produced under budget limits, they are said to be feasible. It's important to estimate the cost of software creation before creating a product. Practitioners have voiced concern about their failure to predict software development costs correctly. If the cost of construction continues to rise, this issue has become much more urgent. As a result, much research is now focused on improving our understanding of how to evaluate automated cost estimation tools. Centered on the parameters implement capacity, extensibility, durability, and traceability, and techniques used to estimate software costs, this paper summarizes software cost estimation models COCOMO II, COCOMO, PUTNAM, STEER, and ESTIMACS.

Shish & Dharmender(2012), estimating the commitment taken to produce high-quality applications is what software development effort calculation is all about. Since inaccurate estimates create problems during the execution of software processes, effective software production necessitates accurate estimates. As a result, the aim of this paper is to recommend a metric for estimating software development effort (SDE) focused on the difficulty of yet-to-be-developed software requirements. The requirement-based complexity is based on the proposed program's software specifications specification (SRS), which allows for a comprehensive and reliable calculation of SDE. The proposed SDE metric is categorically compared to other existing SDE estimation practices proposed in the past, such as algorithmic models, feature point count, usage case point, and lines of code, for validation purposes (LOC). The findings support the argument that the proposed SDE measure is rigorous and detailed, and that it correlates favorably to other widely used SDE measures. In comparison to other software development initiative activities suggested in the past, it is much more valuable because the complexity and SDE figures are collected at a very early stage of the software development life cycle (SDLC).

3. Reviews on software estimation models

Here is a brief of Software Effort Estimation to provide background and existing awareness of the effort estimation methods. It is hoped that by critically discussing all of the major methods proposed from the mid-1970s to the present, the important themes and innovations that have influenced estimation research and practice would be revealed.

According to the existing literature, software estimation models can be broadly categorized into two types (Leung & Fan 2002; Tronto et al. 2008):

(i) Algorithmic Models

(ii) Non-Algorithmic Models

Many researchers have worked over the last two decades to develop models based on case-based reasoning (Finnie et al. 1997), classification and RT (Srinivasan & Fisher 1995), simulation models (Shepperd & Kadoda 2001), neural networks (Park & Baek 2008), Bayesian statistics (Jorgensen 2007), lexical analysis of requirement specifications (Jorgensen & Shepperd 2007), genetic algorithms (Jorgensen (Jorgensen 2007). It has also been designed to combine two or more of these models.

Since expert estimation techniques have been widely accepted by software professionals, the majority of research in the last decade has been based on expert estimation. Best practice guidelines were invoked in the article Jorgensen (2004) to explain what they are like and what the possibilities are for implementing them in software organizations. Jorgensen and Jorgensen-Herr (2007) report on the expert judgment-based SEE guidelines, as well as the two key expert judgement and structured model guidelines.

Tan et al. (2012) proposed a hybrid method for rule learning, induction, selection, and extraction in FRBSs, with the model combining FRBS, GA, and expert judgment using the Pittsburgh approach. The expert estimation is combined with a soft computing method, such as a neural network, by choosing FP and six inputs from the expert estimation, and the use of the neural network technique results in error reduction (Park & Baek 2008). The expert estimation combined with the preparation poker methodology focuses primarily on group discussion and information sharing (Ostvold et al. 2008). It asserts that the precision of the group estimate is greater than that of the individual estimate, and that code analysis further improves code quality because the planning poker is primarily used in agile (Mahnic & Hovelja 2012).

Researchers focused on feature selection in effort estimation to improve model efficiency and reduce model complexity. Fuzzy logic algorithms that use function subset selection techniques are being designed to improve the accuracy of the analogy SEE model (Azzeh et al. 2008). The degree of similarity between two given fuzzy clusters, and thus the similarity between all clusters for all selected features, is evaluated. By ranking the features with respect to weights, feature weighting heuristics for analogy-based effort estimation models are proposed (Tosun et al. 2009). The heuristic is used to change the normal method of Principal Components Analysis (PCA), and the function ordering

corresponds to the eigenvalues and eigenvectors. While accuracy is enhanced for particular projects, it does not include all types of projects in order to highlight accuracy. For effort prediction, ARM and associative classification are also widely used (Alvarez 2003; Christopher 2011). The "support" or "trust" and "knowledge gain" that comes with each decision is used as the main component for filtering all of the laws. A rating algorithm rule based on ordered attributes assesses the rule to be rated's condition, achievement, mastery, and understanding. A rule priority algorithm builds on other rules to find the rule with the highest priority for the learner. A rule sorting system arranges the rules based on the accuracy of their predictions. The hill-climbing method (shown in the chapter) is used to determine the threshold value for support and trust, which is then used to filter the association rules (Coenen & Leng 2007). Fuzzy systems, at their heart, are decision procedures that require the use of fuzzy membership functions, which describe multiple membership functions that reflect the degree of membership of a particular element into a class. To arrive at an optimal solution, these fuzzy membership functions can be linked together using a combinatorial optimization technique. Ishibuchi et al. suggested a classification scheme based on fuzzy if-then rules for multidimensional pattern classification problems with several attributes using a Michigan and Pittsburgh approach (1997). GA computes the comprehensibility, J-measure, and mathematical motion as a multi objective function after extracting a collection of rules from a data set (Wakabi-Waiswa & Baryamureeba 2008). Another research (Shippert & Macdonell 2012) that used a prediction method to estimate a software project is also mentioned. According to Song et al. (2006), high support and confidence levels do not result in higher prediction accuracy, and a sufficient number of rules is needed for high prediction accuracy during software defect association mining. When images are batched together, the average RT rate and width are satisfactory (Minku & Yao 2013). In a series of tests, Menzies (2013) compared four types of miners. Kocaguneli et al. (2012) proposed that dynamic selection of nearest neighbors would boost estimation of nearest neighbors. Analogy-based Software Development Effort Estimation (ASEE) techniques have also received a lot of attention in the last few years (Idri et al. 2015).

3.1 Algorithmic Cost Estimation Models

According to Tronto et al. (2008), Calibration is a requirement for algorithmic cost estimation. This algorithmic cost estimation model's main goal is to establish a relationship between one or more variables. The primary effort driver for this model is software scale. The size of software is proportional to its source LOC, FP, and use case points.

The SLIM, COCOMO, and Jensen models are the most well-known algorithmic estimation models. SLIM employs the Rayleigh curve model, which is the most complex of the three. The cost drivers have an effect on COCOMO. The features of a product are the primary determinants of production cost and timeline.

- **SLIM Model**

Putnam (1978) applied the Rayleigh-curve into software development projects. He examined a military project which is related to very large DB. Then he proposed a estimation tool called SLIM. The equation related to this model is defined as,

$$S_s = cK^{1/3}T^{4/3} \quad \text{.....(2.1)}$$

where S_S is measured in LOC. The constant c is a technology factor that takes into account the effect of numerous productivity related aspects of the project such as complexity. K represents the total life-cycle effort excluding requirement specification and T is the development time measured in years. The constant c, described by Putnam as a "funnel through which all system development must pass", can be assigned one of the 20 values in the range between 610 and 57,314 and as c increases so does productivity. The estimator also has control of the slope of the curve using what is known as the MBI. The higher the mean build-up index (MBI), the more people will be working on the project at the start of it. Rayleigh-curve models give great attention to the relationship between development time and has been due to effort time. Timing is everything when it comes to effort. If you give a little more effort at the beginning, it provides a lot more at the end. This is true for time as well. The more time you can give at the beginning, the more time you get at the end (in terms of effort). By rewriting the above equation, we can find out how long the breath hold an athlete can hold.

$$K = \left(S_s / c\right)^3 / T^4 \quad \text{.....(2.2)}$$

COCOMO Model

TBoehm (1981) developed the COCOMO model, which is a well-known empirical estimation strategy in algorithmic models. The COCOMO model yields three major types of estimation: effort time series, cost time series, and plan time series. The algorithm, which is built on a parametric model, yields even better results. The method may be implemented at various stages, which imply the degree of precision with which the average costs are calculated. The biggest disadvantage of this model is that it is slightly flawed. An inaccuracy in the data is not always the responsibility of the reviewers or project team, but can be due to issues such as hardware specifications, insufficient resources, and equipment, all of which can restrict the reliability of the research findings. COCOMO is a size-based model that uses project sizes such as LOC, WLOC, VLOC, and MLOC. When COCOMO cannot be calibrated, it can still provide predictions about the software's development environment. Consider the following: The intermediate COCOMO model estimates the effort by adjusting 15 cost drivers. The knowledge defines detailed COCOMO as an extra level of detail that is offered. To achieve more reliable estimation, Boehm proposed that various levels of the project hierarchy be handled differently. If you're interested in technology, there are many fascinating project hierarchies to discover from examples such as module, subsystem, and framework.

In COCOMO 81, effort is calculated by using the following formula.

$$PM = a \times size^b \times \prod_{i=1}^{15} EM_i \quad \text{.....(2.3)}$$

Here "*a*" and "*b*" are the domain constants which is included in the estimation model. EM is also used for calculating the effort. COCOMO estimation model collects the experience and data of the past projects, which is especially complex to understand and apply the same. The right hand side of the equation is denoted as PM. The following details are related to the evolution of COCOMO model.

COCOMO II was the uniform for all SEEs in 2000. It examines not only the SDLC steps, such as specifications, but also the design, implementation, and testing steps. COSYSMO 2002, in addition to a current cost model, is a theoretical application of systems engineering. The numerous software development activities that fully estimate the system engineering effort associated with software design, deployment, and testing. The concept behind i-DAVE 2003 was to include facts to address the question of what dependability entails. In 2004, a series of derivative models known as COINCOMO and DBA COCOMO were developed. COINCOMO stands for Constructive Incremental COCOMO. Since everything appears to be a system, a Constructive System-of-Systems Integration (SSCI) model was created to estimate the effort required to integrate software system components. To improve the cost effectiveness of a security supply system, the Constructive Security Cost model (Colbert 2006) was created. It all comes down to behavioral analysis. The COSMIC 2007 software effort calculation metric estimates the functional scale of a software development project. The Constructive COTS model is used to estimate the evaluation, tailoring, and integration costs. This demonstrates the ability to calculate different costs and benefits. A COTS system is evaluated by choosing from a larger system. Any of the projects related to the development of the COTS program are referred to as "tailoring" projects. The Constructive Rapid Application Development Model arose from the realization that development costs are still excessively high as compared to similar development models. Qin and Fang (2011) listed the COCOMO2 model, which makes it more adaptable to modern software industry growth. With the rapid development of information technology, more new optimization approaches will emerge. For a COPLIMO model, the costs of creating a new software object which often involve the cost of additional work required to reuse the object as the software system is further maintained over time. Many who are concerned about the cost of a project also concentrate on the quality cost.

COCOMOII has two types of models: Early Design Model (EDM) and Post-Architecture Model (PAM). EDM is described as a high-level model used to assess the prospects of incremental development strategies. EDM is favored during the early stages of production when accurate knowledge is unavailable. The detailed knowledge is about actual software and the description of the built process, primarily the entire architecture.

PAM's primary working condition is a detailed requirement, and the program is ready for production. As a result, PAM can be built using a fielded framework. PAM has extensive information on metrics such as cost generators, which it uses to generate improved cost estimates. Both EDM and PAM use product sizing and scale factors in the same way. The PAM model is made up of 17 EM and 5 scale factors. Cost drivers have a rating standard that expresses the driver's effect on the production effort, PM. The COCOMO model is a parametric model that was developed using data from a large number of software projects.

3.2 Non-algorithmic cost estimation models

➢ **Expert Judgment**

Jorgenson (2004) suggested the best practice guidelines for expert estimation. The following twelve expert estimation principles are assessed from the review.

(i) Accuracy of estimation is basically related to evaluation measures.
(ii) To avoid inconsistent goals of estimation.
(iii) Collect the positive and negative estimation points from estimators.
(iv) Avoid unrelated and unpredictable estimation information.
(v) Use reported data from existing development tasks.
(vi) Collect appropriate domain knowledge from experts and high quality estimation records.
(vii) Top-down and bottom-up estimation is applied separately to each other.
(viii) Proper checklist is used for estimation.
(ix) Integrate different expert estimation and their estimation strategies.
(x) Evaluate the unreliability of the estimate.
(xi) Give assessment on estimation accuracy and development task relations.
(xii) Estimation training opportunity is essential.

Jorgenson (2004) provides proof for twelve estimation principles as well as recommendations for applying them to the software industry. Expert estimation is more reliable than model estimation because the experts have their own domain expertise. Model estimation is more reliable than expert estimation because the experts do not have their own domain expertise. When experts use true and easy estimation methods, their estimation performance is excellent or better than model estimation. Otherwise, the strategies could result in skewed estimates. Overestimation results from biased inference, but estimation models are less biased.

Jorgenson (2005) proposed three useful criteria for expert judgment-based SEE: pX effort estimate, expected effort, and effort-to-win. The estimator calculates the effort level with an X percent likelihood of not exceeding the expected effort using pX effort calculation. The primary goal of providing a pX effort estimate is to improve the model's accuracy. For example, if a project leader calculates a project's p50 effort estimate as 20,000 work hours, it means that the above-mentioned situation has a 50% chance of not reaching 20,000 work hours. The expected effort is calculated using the pX effort estimate. When the value for X in pX is set to 80, the expected effort has a low chance of going over budget. A project manager employs the p30 effort estimate because a lower planned effort may result in more productive work. Thus, accuracy, project management, and job productivity are considerations taken into account when determining the expected effort.

In the effort-to-win principle, the client embraces the effort after the bidding round. When evaluating effort-to-win, the primary concern is market price and financial advantage. The pX effort calculation can be used to calculate benefit potential.

The following considerations must be made in order to accept and measure the uncertainty of effort use.

(i) *p*50 effort estimate is the most likely effort estimation.
(ii) Fix the minimum and maximum values of effort. Example: Fix the maximum effort to 150 percent and the minimum effort to 90 percent of estimated effort.
(iii) Planning, budgeting, or other information needs are involved in fixing the values.
(iv) Probability of the estimation then the actual effort falls within the minimum-maximum.

Jorgensen (2007) examined the estimation of software development job effort using expert judgment, a systematic model, and a combination of the two methods. According to the study, expert judgment–based effort estimation outperforms model estimation in terms of accuracy. The primary explanation for this is that formal estimation models are not calibrated in the company, and experts with contextual knowledge are exempt from formal estimation. Four papers are linked to the use of expert judgment and models in combination. The combination-based methods' average estimation accuracy is comparable to that of the other estimation methods.

The design of software development estimation models entails the selection of a limited number of effort-related parameters. The importance of these parameters in estimation models cannot be overstated. A large number of parameters are considered when designing the model, which contributes to overfitting and reduces accuracy. If the model is basic, users will be able to grasp it quickly. The estimation model is built with a large number of parameters. The primary drawbacks are

(i) Software development estimation models are complex.
(ii) It takes much effort for designing the estimation models.

Design factors suggested from the analysis of expert estimation

(i) Selection of appropriate estimation method
(ii) Choosing appropriate estimation models
(iii) Calibration level
(iv) Expert's contextual information is used in the estimation model
(v) Selecting proper expert judgment process
(vi) Choosing expertise of expert judgment estimation
(vii) Mention motivational biases in estimation methods
(viii) Input of estimation
(ix) Contextual Information
(x) Complexity of estimation
(xi) Limitations
(xii) Design issues

***Merits and Demerits**:* Expert estimation process is relatively cheap and the accuracy of results depend on the domain knowledge of the experts. The difficulty with this approach is that information related to existing related projects are not available.

➢ **Analogy Based Estimation**

Azzeh et al. (2008) introduced an analogy approach to software development estimation. In the comparison method, software development effort is measured using historical data from projects similar to the current project. The properties of a dataset have a direct impact on the model's accuracy. Azzeh et al. (2008) examined success in a language discrimination task based on the function subset subsets selected. Techniques for selecting function subsets based on analogy are linked to fuzzy logic. The sentence employs SEE for the term "used," as well as two Desharnais datasets. We may use various feature selection approaches, such as hill climbing, forward subset selection, and backward subset selection.

The best feature subset calculated using Fuzzy Feature Subset Selection (FFSS) in the ISBSG dataset are features such as maximum team size, development type, resource level, and average team size. Using FFSS, the best features in the Desharnais dataset are Team experience, Manager experience, Transaction, Envergure, and Points Non-Adjust.

Tosun et al. (2009) proposed an analogy-based cost estimation approach for estimating software costs. This approach makes use of historical project data. It compares the features of previous and current projects and measures the cost of the current project based on the costs of previous projects. The downside of this approach is that it considers all project features to be equal. These features have an impact on the project's cost. For cost estimation, the model employs two function weight assignment heuristics. The PCA weighting method is used to separate optimal linear patterns from high-dimensional data. Two heuristics are used to rate the functions. It first modifies the standard PCA procedure, which is based on the ordering of eigenvectors and eigenvalues, and then modifies the orderings for features if they are found. According to the findings, the assessment metric MMRE, portfolio size increases more in the Albrecht and Desharnais dataset.

Idri et al. (2015) conducted a study of ASEE papers, with two key goals in mind: The first goal is to apply new or old ASEE techniques for research approaches, contribution sort, techniques used in conjunction with ASEE methods, and ASEE measures, as well as to define publication channels and trends. Explain the process of estimating accuracy, examine the process of estimating accuracy, examine the process of estimating accuracy, and explain what happens in a posteriori estimation. The ASEE techniques are used to discuss feature and case selection, similarity steps, and adaptation strategies. There have been almost as many reviews (18 reviews of 38 articles) of E-cigarette studies as there have been. The most commonly used techniques were SM (18 reviews), Fuzzy Logic and/or Analogy/Evaporative Assumption (18 reviews each), and GA (18 reviews) (8 reviews). One-third of cost estimation approaches are focused on expert judgment (18%), Least Square Regression (LSR) (15%), and Grey Relational Analysis (GRA) (2 percent). (GRA.) I (ii) It does not manage "missing values" and "imprecision" in the estimates, (iii) "Quality of estimates greatly affects quality of the estimates," as discussed in detail, and (iv) The ASEE model is the only one that does not make mistakes when the data are sampled poorly."

Advantages and disadvantages: The precision of estimation in an analogy-driven estimation method is based on knowledge inferred from similar ventures. The disadvantage of using an analogy-based calculation method is that the expense database must be kept up to date in a systematic manner.

➢ **Machine Learning**

Park and Baek (2008) proposed empirically validating a neural network model for SEE. The neural network model's output is influenced by the input variables. Three neural network models are constructed using sufficient input variables, and their output is compared. Three neural network models each use a different set of input variables. Enterprise Miner v 8.2 is used to construct neural network models. The training and validation datasets are drawn from 148 datasets in a seven-to-three ratio.

Heiat (2002) provided a comparison of ANN and regression models for software development estimation. When it comes to this ANN, the MultiLayer Perceptron is a very powerful algorithm. For measuring arbitrary functions, mixing programs are used. MLP was trained using a back propagation algorithm. The neural network uses the Radial Basis Function (RBF) network, which has more benefits than MLP. In SEE, the neural network method is compared to the simpler regression model using multivariate pattern recognition. The neural network model is fed data from a third or foreign language. The neural network model produces more reliable results than the traditional regression algorithm. The third and fourth generation languages are used to train a neural network to predict the native language of the speaker. Even though the neural network model's computed Mean Absolute Percentage Error (MAPE) is lower than the MAPE of a regression model, the network must be trained more than a regression model.

Tronto et al. (2008) suggested researching ANN-based prediction systems in software project management. (Use "Software project management" for simplicity, or "Project management" for a more thorough understanding.) Data manipulation is part of ANN models (ANN stands for "any amount of"). Keeping time, temperature, pitch, and tone constant in n-dimensional space, a manipulated variable (or a note) is then mapped into that unitary space. Normalization is used to reduce the bias of input variables that have large values in comparison to the other variables. There are certain values that must be chosen in order to construct a natural distribution. These values can be chosen by taking the maximum value for each variable and mapping the data to the hyper-spaces provided by the proposed normalisation. There is one hidden layer, two input neurons, and one output neuron in the neural network. The secret layer contains 23 neurons. A logistic sigmoid activation function and a linear activation function are used in the hidden layer. A comparison of ANN to simple linear regression and regression with multiple variables demonstrates that ANN is a superior regression approach to simple LRM and multiple regression.

According to Oliveira et al. (2010), a GA-based approach for learning a machine learning regression on SEE can be used for feature selection as well as parameter optimization. The main goal of this research is to select an optimal subset of features from web pages, optimize the features that we select, and use machine learning algorithms on the framework to improve SEE accuracy.

The original researcher of the Oliveira et al. model, Desharnais, NASA, COCOMO, Albrecht, Kemerer, and Koten datasets, and Gray datasets are used for evaluation to gain insight and interpretation of the model proposed by Oliveira et al (2010). On the aforementioned datasets, we are comparing neural networks, support vector machines, multiple additive regression, bagging, and Bayesian statistical models to the GA-based estimation process. The GA-based approach outperforms neural networks, support vector machines, multiple additive regression trees (st04824), bagging (st4823), Bayesian statistics models (st4822), and normalised support vector machines on the six datasets (st4821).

Tan et al. suggested a hybrid approach to rule learning in FRBSs that incorporates rule induction, rule extraction, rule selection, and rule learning. As a result, the model mixes "Freakonomics" and "Game Theory." A FRBS should include a fuzzification module, a defuzzification module, a Knowledge Base (KB), and a decision making unit. The KB is a compilation of DB and RB. A rule base begins by defining Membership Functions (MF), which determine which set of values the rule attaches to. Then, using fuzzy laws, decision units perform inference operations on related values based on the MF's outcome. The fuzzification module will receive inputs, while the defuzzification module will translate the fuzzy output into an informative and understandable format for humans.

Attarzadeh and Ow (2010) proposed a novel soft computing-based algorithmic cost estimation model. Fuzzy logic is a newer soft computing method for estimating effort based on non-algorithmic techniques. It is referred to as a strong linguistic representation that can reflect imprecision in inputs and outputs, resulting in a model with a more knowledge-based approach. As compared to the neural network model, the fuzzy logic model performed well.

Martin and colleagues contrast FLM with the Linear Regression Model (LRM) (2008). As measurement metrics, the Magnitude of Error Relative (MER) to the estimate and the Mean of MER (MMER) are used. Three FLM are generated using a triangular fuzzy membership function, a trapezoidal membership function, and a Gaussian membership function. Data is fed into the three FLMs through twenty programs written by seven programmers.

To implement a fuzzy scheme, the various input categories must be represented by fuzzy sets, which are then represented by MF (Ahmed et al. 2005). The fuzzy logic model is fed a variety of inputs, and the MF parameter values are defined using an interval. The smallest MMRE is obtained by iteratively adjusting intervals from MF values equivalent to or equal to both the minimum and maximum of program sizes and efforts. When the FLM and LRM models are compared, the FLM model outperforms the LRM model in terms of predictive accuracy.

A Compact Fuzzy Association Rule-Based Classifier was proposed by Pach et al. (2008). (CFARC). This designation employs a well-known data mining methodology known as ARM. The $X \Rightarrow Y$ inference is a common type of association law. The association law is made up of an antecedent (X) and a consequent (Y). Associative classification is used to connect the discovery and classification of associations. The primary drawbacks of associative grouping are as follows:

(i) huge amount of rules,
(ii) problems in fixing values for support and confidence, and
(iii) continuous data handling

Pach et al. created a novel fuzzy rule-based classification approach to address the above issues (2008). The two problems in fuzzy modeling that are maintained in the proposed approach

are interpretability and accuracy. The findings show that the classification model functions effectively when discovered by a compact set of classification rules with interpretability, and that the classification model's accuracy increases. The partitioning of continuous (numerical) attributes influences the classification model's accuracy.

Pach et al. (2008) proposed the CFARC model, which produces very compact (low number of rules) and more accurate classifier systems.

Shepperd and Macdonell (2012) suggested a new method for testing prediction systems in software project estimation, which was presented. Metrics such as SA, random guessing, and effect size are calculated using this method.

According to Shepperd and Cartwright (2005), the estimation scheme produces results that are no better (in reality, they are worse) than random guessing. Shepperd and Schofield (1997) contrasted the results of two prediction methods, EBA and Step Wise Regression (SWR) analysis, and referred to the evaluation metric MMRE as an unsafe scale (1997). According to Shepperd and Macdonell (2012), the MMRE of PEBA should be lower than that of PSWR, with PEBA referring to the MMRE of EBA prediction and PSWR referring to the MMRE of SWR research prediction, and the MAR of PSWR should be lower than that of PEBA. When compared to random guessing, SA for two approaches improves performance by 50%. If the effect size values obtained are small or medium, the model is statistically important. The SA metric is a ratio that represents how much better the Prediction Method (Pi) is compared to random guessing. The SA assessment metric is used in the proposed framework defined in this thesis.

Song et al. (2006) predicted defect associations and the effort needed to correct them using ARM-based methods. The system can be used by software developers to identify software bugs, and project managers can use the data to enhance software management and testing resources. The model is validated using defect data from the Software Engineering Laboratory (SEL), which covers more than 200 projects over a 15-year period. According to the results, the accuracy of defect association prediction is very high, and the false-negative rate is very low. According to the results for defect correction effort prediction, the accuracy for both defect isolation effort prediction and defect correction effort prediction is high. The values of support and confidence levels influence prediction accuracy as well as the number of false positives.

There are many variables to consider, including the number of laws, the negative rate, and the rate of false positives. In ARM, rules that go beyond the bare minimum to support the rules are normal. Quite strict laws have a higher level of trust than the minimum level of confidence. The ARM consists of two stages, which are as follows:

(i) Finding the frequent item sets
(ii) Generate strong rules

The length-first rule-ranking strategy is as follows.

(i) Based on their length rules are ranked. High priority is given to the longer rules.
(ii) If Length of the two rules is same, the rules are ranked according to their confidence values. Highest priority is assigned to rule with highest confidence value.
(iii) If confidence of the two rules is same, the rules are ranked according to their support value. Highest priority is assigned to rule with highest support value.
(iv) If two rules have the same support value, they are ranked in alphabetical order.

The three components of prediction accuracy P are formula accuracy, false-negative rate, and false-positive rate. The defect correction effort prediction method is compared to the PART, C4.5, and Naive Bayes methods. The defect isolation effort improves accuracy by 25% as compared to the other three approaches. When the three methods for predicting defect correction effort are compared, the accuracy increases by 23%. The most significant discovery is that higher levels of support and confidence do not always mean higher prediction accuracy, and that a sufficient number of rules is needed for high prediction accuracy.

Minku and Yao (2013) suggested an ensemble data mining model for the SEE. The term "ensemble data mining" refers to the use of a built model in conjunction with machine learning methods. It outperforms a single model in terms of accuracy. The following objectives are pursued by these ensemble methods:

(i) Evaluation of existing ensembles of learning machines to improve SEE. Compare Ensemble method with single learning machines and identify which is useful
(ii) Examining the adequacy of different locality approaches
(iii) How to improve SEE?
(iv) How to evaluate/select machine learning models for SEE?

In terms of performance across various data sets, RT bagging ensembles are highly rated.

Kultur et al. (2009) used bagging in an adapted version. According to the results, five data sets produce very significant improvements in bagging as compared to single learning machines. The parameters used have an impact on the machine learning experiments. Depending on the parameter chosen, an approach will improve or deteriorate. Three ensemble learning machines and three single learning machines were used in the study.

Merits and Demerits: Appropriate choice of parameters using machine-learning algorithms for calculation of effort results in improved accuracy of the model. However, rule optimization should be done for improving the results obtained.

➢ **Hybrid Techniques**

Araujo et al. (2012) suggested a hybrid morphological approach for estimating software development costs. A hybrid methodology is used in this paper to construct Morphological-Rank-Linear (MRL) perceptrons for the SDCE problem. The parameters of the MRL perceptron are optimized using adjusted GA (MGA), which also chooses an optimal input function subset from the data bases used. The MRL perceptron parameters supplied by the MGA are improved using a gradient steepest descent method for each MGA individual. Six well-known benchmark data bases of software projects were used for the experimental research. To evaluate its results, the hybrid model employs the evaluation metrics MMRE and PRED (25) Evaluation measures and

fitness functions are used to evaluate the hybrid model's results. A Morphological-Rank-Linear Hybrid Design is used to estimate SDCE (MRLHD). It uses an evolutionary search technique to find the elements needed to construct an accurate method (Leung et al. 2003).

i) Model structure, parameters, and training algorithm (ii) Cost estimation details (the minimum dimensionality, or number of features, for representing the data).

Some of the datasets used to evaluate the hybrid model include Nasa, Desharnais, COCOMO, Albrecht, Kemerer, and Koten- Gray. The proposed model is compared to current models such as SVR-Linear, SVR-RBF, Bagging, GA-based with SVR-Linear, GA-based with SVR-RBF, and MRL using two output tests, MMRE and PRED (25) When compared to existing approaches, the hybrid model yields better performance. It indicates a 0.6 percent improvement for Nasa DB, an 11 percent improvement for Desharnais DB, a 12 percent improvement for COCOMO DB, an 8 percent improvement for Albrecht DB, a 12 percent improvement for Kemerer DB, and a 0.2 percent improvement for KotenGray DB as compared to the best results published in the current literature.

According to Idri et al., missing data (MD) in a dataset may have an effect on software effort prediction systems (2016). An MD technique is used to estimate software development effort using two classical analogies and a fuzzy analogy. Seven continuous datasets are being considered for this project. The three missingness mechanisms are missing entirely at random (MCAR), missing at random, and Non-Ignorable Missing (NIM), and the three MD techniques used on seven datasets are toleration, deletion, and K-Nearest Neighbors (KNN) imputation. The accuracy of the estimation model decreases as the percentage of missing data in the dataset increases, according to the findings. It also states that the fuzzy analogy model outperforms the classical analogy model in all datasets. In terms of precision, the KNN imputation technique outperforms MD toleration and deletion techniques. In this analysis, the model is only evaluated with continuous datasets, which is a limitation.

4. Conclusion

In recent decades, a variety of models and methods for estimating effort have been developed. This shows that the researchers are aware of the need to enhance effort estimation in software engineering. The software development process is influenced by a variety of factors. These factors can be human or technological, and their consequences are never completely predictable. In this paper, we looked at various estimation techniques and illustrated methods for measuring size during the estimation process. If the estimation is performed correctly, the error will be reduced. The estimation method represents the current state of the project. The measurement of effort is still inaccurate. To overcome this inability, a variety of techniques have been developed, including the use of case points, story points, and so on. We may compare the ability of these techniques in future work.

References

1. Azzeh, M, Neagu, D & Cowling, P 2008, ‘Improving analogy software effort estimation using fuzzy feature subset selection algorithm’
2. Tosun, A, Turhan, B & Bener, AB 2009, ‘Feature weighting heuristics for analogy-based effort estimation models’, Expert Systems with Applications, vol. 36, no. 7, pp. 10325–10333.
3. Idri, A, Amazal, FA & Abran, A 2015, ‘Analogy-based software development effort estimation: A systematic mapping and review’, Information and Software Technology, vol. 58, pp. 206–230
4. Heiat, A 2002, ‘Comparison of artificial neural network and regression models for estimating software development’, Information and Software Technology, vol. 44, no. 15, pp. 911–922.
5. Tronto, IFB, Silva, JDS & Anna, NS 2008, ‘An investigation of artificial neural networks based prediction systems in software project management’, Journal of Systems and Software, vol. 81, no. 3, pp. 356–367.
6. Pach, FP, Gyenesei, A & Abonyi, J 2008, ‘Compact fuzzy association rule based classifier’, Expert Systems with Applications, vol. 34, no. 4, pp. 2406–2416.
7. Shepperd, M & Cartwright, M 2005, ‘A replication of the use of regression towards the mean (R2M) as an adjustment to effort estimation models in software metrics’, Proceedings of the eleventh IEEE international software metrics symposium, pp. 10−38
8. Song, Q, Shepperd, M, Cartwright, M & Mair, C 2006, ‘Software defect association mining and defect correction effort prediction’, IEEE Transactions on Software Engineering, vol. 32, no. 2, pp. 69–82.
9. Martin, CL, Marquez, CY & Tornes, AG 2008, ‘Predictive accuracy comparison of fuzzy models for software development effort of small programs’, Journal of Systems and Software, vol. 81, no. 6, pp. 949–960.

