Mehran University Research Journal of Engineering and Technology
Vol. 39, No. 2, 413- 431, April 2020
p-ISSN: 0254-7821, e-ISSN: 2413-7219
DOI: 10.22581/muet1982.2002.18

A Study of Software Development Cost Estimation Techniques and Models

Junaid Rashid[1], Muhammad Wasif Nisar[2], Toqeer Mahmood[3],

Amjad Rehman[4], Syed Yasser Arafat[5]

RECEIVED ON 17.10.2018, ACCEPTED ON 03.05.2019

ABSTRACT

SDCE (Software Development Cost Estimation) has always been an interesting and budding field in Software Engineering. This study supports the SDCE by exploring its techniques and models and collecting them in one place. This contribution in the literature will assist future researchers to get maximum knowledge about SDCE techniques and models from one paper and to save their time. In this paper, we review numerous software development effort and cost estimation models and techniques, which are divided into different categories. These categories are parametric models, expertise-based techniques, learning-oriented techniques, dynamics-based models, regression-based techniques, fuzzy logic-based methods, size-based estimation models, and composite techniques. Some other techniques which directly do not lie in any specific category are also briefly explained. We have concluded that no single technique is best for all situations; rather they are applicable in different nature of projects. All techniques have their own pros and cons and they are challenged by the rapidly changing software industry. Since no single technique gives a hundred percent accuracy, that is why one technique and model should not be preferred over all others. We recommend a hybrid approach for SDCE because in this way the limitations of one model and technique are complemented by the merits of the other model/technique. We also recommend a model calibration to obtain accurate results because if a model was developed in a different environment, we cannot expect reliable estimates from it in a completely new environment.

Keywords: **Cost Estimation, Cost Estimation Models, Effort Estimation Techniques, Software Development Cost Estimation.**

1. INTRODUCTION

SDCE is a primary activity in project management to manage resources in an effective way by predicting the required amount of effort to fulfill a given task [1]. The accurate effort estimates, which have a major impact on software development management, increase the chances of accomplishing the required work related to a software project within time and budget. If the manager's estimate is too low, it will cause a

[1] Department of Computer Science, Air University Islamabad, Kamra Campus, Pakistan.
Email: junaidrashid062@gmail.com (Corresponding Author)
[2] Department of Computer Science, COMSATS University Islamabad, Wah Campus, Pakistan.
Email: wasifnisar@gmail.com
[3] Department of Computer Science, National Textile University, Faisalabad, Email: toqeer.mahmood@yahoo.com
[4] Artificial Intelligence and Data Analytics Lab (AIDA), Prince Sultan University, Riyadh, Saudi Arabia.
Email: drrehman70@gmail.com
[5] Department of Computer Science, University of Engineering and Technology, Taxila, Pakistan.
Email: syed.yasser.arafat@gmail.com

development team to be under time pressure, leading to residual errors due to incomplete software functionality and insufficient testing. Conversely, if the manager's estimate is too high, development resources and personnel will be overly allocated to the project, making a company failed to secure a contract due to noncompetitive contract bids. The software quality is the essential thing for an organization which depends on customer satisfaction with product and requirements [2, 3].

The existing software cost estimation models and techniques are divided into six major categories: parametric models, learning-oriented techniques, expertise-based techniques, regression-based models, dynamics-based models, and composite-bayesian techniques [4]. Much research has been carried out that increasing demands of high-quality software through effective cost estimation. The vital issue which is closely related to the software projects financial aspects is the accurate estimation of software cost. This will help with the management of software projects budgets. There is a relationship between the estimation of software and the cost of software, so it is said that the primary factor for software cost is an effort. Software estimation supports in fixing the exact targets in software project completion [5].

Several research studies are carried out for surveying effort and cost estimation techniques [6]. To the best of our knowledge, none of them cover all the SDCE methods. They only discuss popular techniques i.e. COCOMO (Constructive Cost Model) [7], function points, SLIM, Delphi, NN (Neural Networks), regression-based techniques, etc. So, there is a need to gather a maximum number of classical as well as the latest techniques in one paper and to provide their overview. Software models have utilized in model-driven engineering [8]. These techniques are tabulated category wise in Table 1. We do regard the two terms of cost and effort interchangeably in this work.

This study is undertaken to study the utmost number of SDCE techniques. The main objective of this paper is to support software development effort estimation by exploring and collecting SDCE techniques in one place. This contribution in the literature will assist future researchers of SDCE to get maximum knowledge about SDCE techniques from one paper. The remainder of this paper is organized as follows: Section 2 describes the parametric models for cost estimation; the expertise-based techniques are presented in Section 3, learning-oriented techniques are discussed in Section 4, dynamics-based models are given in Section 5, regression-based techniques and fuzzy logic based methods are presented in Section 6 and 7 respectively, size-based estimation techniques and composite techniques are discussed in Section 8 and 9 respectively, Section 10 describes some other techniques, Section 11 contains the discussion, and Section 12 concludes the paper.

2. PARAMETRIC/ALGORITHMIC MODELS

Algorithmic models are those which generate cost estimates as a function of major cost factors. If E denotes effort, and CF denotes cost factors, then the algorithmic model will get the form as in Equation (1).

$$E = f(CF_1, CF_2, CF_3, \cdots, CF_n) \tag{1}$$

All the existing algorithmic methods have their own way of selecting cost factors and the form of the function f. Parametric models "calibrate" pre-specified formulas for SDCE from historical data [5]. The estimation is not based on the analysis of tasks, but they take counts of inputs and then generate outputs.

2.1 Software Life-Cycle Model (SLIM)

A software lifecycle model also called the putnam model, is an empirical software effort estimation model developed [9]. SLIM takes SLOC (Source Line Number Code) as input to estimate software cost. The basis of this model was putnam's life cycle analysis [10]. Since this model assumes that resource consumption will change over time, it can be modeled by a well-known Rayleigh distribution of project personnel levels over time. The PNR (Putnam Norden Rayleigh) curve formula shows the correlation between the project's application effort and the delivery date. Table 1 shows the categories of the SDCE method.

TABLE 1 . CATEGORIES OF SDCE TECHNIQUES

No.	Technique	No.	Technique	Class/Category
1	SLIM	2	COCOMO	Parametric models / Algorithmic models / Model-based / Knowledge-based techniques
3	SEER-SEM	4	Checkpoint	
5	ESTIMACS	6	PRICE-S	
7	COSYSMO	8	COCOMO-II	
9	Delphi technique	10	Wideband Delphi technique	Expertise-based / consensus-based techniques
11	Work Breakdown Structure	12	Rule-based Systems	
13	Planning Poker	14	Top-Down	
15	Bottom-Up			
16	Case-based Reasoning	17	Neural Networks	Learning-oriented techniques / Machine Learning method / Evolutionary computing
18	Genetic Algorithms (GA)	19	Genetic Programming (GP)	
20	System dynamics approach			Dynamics-based techniques
21	“Standard” regression	22	“Robust” regression	Regression-based techniques
23	Fuzzy Systems			Fuzzy logic / Soft computing based
24	Function Points	25	Full Function Points	Size-based estimation techniques
26	Use Case Points			
27	Bayesian approach			Composite techniques
28	Price-to-win	29	Parkinson	Other Techniques
30	Proxy-Based Estimating (PROBE)			

The Rayleigh manpower equation which is used to derive the software equation is stated in Equation (2).

$$\frac{dy}{dt} = 2Kate^{at^2} \tag{2}$$

In equation (2), K represents the area under the curve, the parameter a = $(1/2t_d^2)$, and $t_d$is the time at which dy/dt is at peak. As parameter K and parameter t_d which affects the value of *a* can take on multiple values, as a result, the Raleigh curve gets different shapes and sizes. Now, putting the value of a = $(1/2t_d^2)$ in Equation (2), the Rayleigh equation will get the form of equation (3)

$$\frac{dy}{dt} = 2K\left(\frac{1}{2t_d^2}\right)te^{-\left(\frac{1}{2t_d^2}\right)t^2} \tag{3}$$

After simplification, the above equation may be expressed as

$$\frac{dy}{dt} = \frac{K}{t_d^2}te^{\frac{-t^2}{2t_d^2}} \tag{4}$$

The basic equation for estimation in the SLM method is shown in equation (5):

$$dt = C_k K^{\frac{1}{3}} t_d^{\frac{4}{3}} \tag{5}$$

Equation (5) can also be written as Equation (6) as

$$S = PP(E/B)^{\frac{1}{3}} t_d^{\frac{4}{3}} \tag{6}$$

In Equations (4-5), C_k is the state of technology, K is applied effort, td is total development time (in years),

S is the size of software in SLOC, PP symbolizes productivity parameter, E represents an effort in person-years, B means a skill factor (a function of software size).

2.2 COCOMO

The COCOMO’81 was developed in 1981, and the software project effort, cost, schedule estimate introduced as a software model [10]. COCOMO'81 was derived from the analysis of 63 software projects completed at TRW Aerospace. There are three levels (sub-model) in Kokomo: Basic Kokomo, Intermediate Kokomo, and Detail Kokomo.

Basic COCOMO calculates labor (man months) and costs as a function of the program size described in thousands of estimated delivered source instructions (KDSI). The basic COCOMO Equation (7) describes.

$$\mathrm{MM} = \mathrm{a(KDSI)^b} \quad (7)$$

The equation (8) is used to get development time.

$$\mathrm{t_{dev}} = \mathrm{c(MM)^d} \quad (8)$$

In Equations (7-8), *MM* is man-month / person-month or staff-month i.e. effort of one person in one month, *KDSI* is a measure of size (length measure) i.e. the number of thousand delivered source instructions (one SLOC may be several DSI), t_{dev} is development time, the values of a, b, c, and *d* are dependent on the mode of development and can be taken from **Table 2.**

Intermediate COCOMO considers those cost factors that were missing in basic COCOMO. It calculates the effort as a function of program size and the four cost consist of a subjective assessment of products, personnel, hardware, and projects, each with several additional attributes, a total of 15 cost drivers/attributes [11]. The intermediate COCOMO equation takes from Equation (9).

$$\mathrm{MM} = \mathrm{a(KDSI)^bC} \quad (9)$$

In equation (9), MM is man-month, KDSI is the number of thousand delivered source instructions, the coefficient *a* and exponent *b* depend upon the mode of development and their value can be taken from Table 2. The *C* is effort adjustment factor which is calculated by multiplying the values of the cost parameter. The development time t_{dev} is calculated the same way as in Basic COCOMO.

Detailed COCOMO, along with the assessment of the impact of each cost factor at each stage of the software engineering process, fits all the functions of intermediate COCOMO. In order to calculate detailed COCOMO man-hour, the entire software project is divided into different modules and man-hours are estimated by applying COCOMO to each module [12]. After that, combine the estimated man-hours of each module to obtain the total man-hour.

2.3 SEER-SEM

System evaluation and resource estimation product called software estimation model (SEER-SEM). It is a software project estimation model which is commercially available and most of its inner details are proprietary. SEER-SEM began with the Jensen model and it estimates effort, cost, risk, and schedule of a project while covering all phases of SDLC. The effort is calculated using Equation (10) [13].

$$\mathrm{K} = \mathrm{D}^{0.4} \times (\frac{\mathrm{S_e}}{\mathrm{C_{te}}})^{1.2} \quad (10)$$

In Equation (10), D is staffing complexity (rating of the project’s inherent complexity with regard to the rate at which staff is added to a project), S_e represents effective size which is commenced earlier, C_{te} indicates effective technology i.e. a metric which contains efficiency factors or productivity factors for carrying out the development process.

2.4 Checkpoint

A checkpoint tool, a knowledge-based software project estimation tool, was developed by software productivity research (SPR) [14]. It has its own proprietary database of thousands of software projects. Use functional points as the main measure of size. Checkpoint tool moves beyond the project-level and phase-level estimation; rather it estimates the effort at activity-level and task-level.

Checkpoint supports the entire SDLC by focusing on three major capabilities i.e. estimation, measurement, and assessment.

TABLE 2 . MODES OF DEVELOPMENT AND VALUES OF COEFFICIENTS FOR BASIC AND INTERMEDIATE COCOMO

Software Project Development Mode	Project Characteristics				
	Size (a)		Innovation (b)	Deadline/Constraints (c)	Development Environment (d)
	Basic	Intermediate			
Organic	Small (2.4)	Small (3.2)	Slight (1.05)	Not tight (2.5)	Stable (0.38)
Semi-detached	Average (3.0)	Average (3.0)	Average (1.12)	Average (2.5)	Average (0.35)
Embedded	Large (3.6)	Large (2.8)	Huge (1.20)	Tight (2.5)	Complex hardware / customer interfaces (0.32)

2.4.1 Estimation

Project, phase, activity, and task are the four levels of granularity at which checkpoint effort. Other than effort, estimates of cost, deliverables, resources, defects, and schedules are also being predicted.

2.4.2 Measurement

Users can execute benchmark analysis, make out best practices, and develop internal estimation knowledge-bases by capturing project metrics using the checkpoint.

2.4.3 Assessment

The comparison of actual performance and estimated performance to different industry standards contained in knowledge-base is facilitated by Checkpoint. Checkpoint also evaluates the strong points and weak points of the software environment.

2.5 ESTIMACS

Originally developed as QUEST (Quick Estimation System), it was integrated into the product line of management and computer Services (MACS) as ESTIMACS. It focuses on the software development phase of SDLC. Since ESTIMACS is a proprietary model, internal details such as expressions being used cannot be used.

This model consists of five sub-models: system development effort estimate, staffing and cost estimate, hardware configuration estimate, risk estimate, and portfolio analysis. These sub-models are used sequentially so that the output from one sub-model is often the input to the next sub-model.

ESTIMACS does not require size measurements about SLOC as input. Rather, it depends on scales like function points for size input. There are 25 input-like parameters in a format resembling questions, which are partly related to the size and complexity of the software you develop, and the complexity of organization and users need to provide answers to those questions [15].

2.6 PRICES-S

The model PRICE-S (Parametric Review of Information for Cost Accounting and Evaluation - Software) was originally developed at Radio Corporation of America (RCA). Since then, it was released as a unique model in the 1970s, so its internal equation was not announced. This model is presented to the user as a black box because its basic concept is not known in the public domain. The PRICE-S tool is currently sold by private company PRICE - Systems. To estimate the relationship between development effort distribution and calendar time, this model uses a 2-parameter beta distribution instead of a Rayleigh curve.

The PRICE - S model consists of three sub-models, an acquisition sub-model, a sizing sub-model, and a life cycle cost sub-model. The acquisition sub-model predicts the cost and schedule of the software, the sizing sub-model helps estimate the size of the software, the life cycle cost sub-model estimates the cost of the maintenance and support phases and is used in combination with the acquisition sub-model.

2.7 COSYSMO

The constructive system engineering cost model (COSYSMO) is a parametric model [16]. COSYSMO is the newest member of the COCOMO family of software cost estimation models [17]. It estimates the effort and time required to perform system engineering tasks. This model contains 14 effort multipliers and 4 size factors, a total of 18 parameters. For sizing, it uses those metrics which are at a level of the system, incorporating both software and hardware.

The COSYSMO went through three main iterations i.e. Strawman COSYSMO, COSYSMO-IP, and COSYSMO. Strawman COSYSMO was the first major version of COSYSMO having 16 cost drivers; half of them were labeled as team factors and the remaining half were labeled application factors. Function points and use cases were the functional size measure. COSYSMO-IP is known as the second spiral of COSYSMO derivation containing a revised set of cost drivers. A general form of the model, mentioned in equation 11, was proposed which contains three parameters of different types i.e. additive, exponential, and multiplicative.

$$\mathrm{PM} = \mathrm{A} \times (\mathrm{Size})^{\mathrm{E}} \times (\mathrm{EM}) \tag{11}$$

In Equation (11), *PM* is effort in person-months, *A* represents calibration factor, *Size* (additive parameter) denotes the measure(s) of functional size of a system having an additive effect on systems engineering effort, *E* (exponential parameter) is scale factor(s) that has an exponential / nonlinear effect on systems engineering effort, and *EM* (multiplicative parameter) indicates effort multipliers which influence systems engineering effort.
The third spiral of COSYSMO derivation is referred to simply as COSYSMO. Using a Delphi exercise by a group of experts, size drivers and cost drivers were determined. The current operational COSYSMO model (central cost estimating relationship or CER) is of the regression equation form, shown as under [18]:

$$\mathrm{PM_{NS}} = \mathrm{A} \cdot (\mathrm{Size})^{\mathrm{E}} \cdot \prod_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{EM_i} \tag{12}$$

In Equation (12), PM_{NS} is effort in Person Months (Nominal Schedule), *A* represents calibration constant (derived from historical project data), *Size* is determined by computing the weighted sum of the 4 size drivers, *E* denotes economy/diseconomy of scale (by default it is 1.0), *n* is a number of cost drivers i.e. 14, and EM_I is an effort multiplier for the ith cost driver.

Modification to the COSYSMO estimating relationship was proposed to remedy some limitations with the current implementation of the cost drivers, by adding two new cost drivers i.e. Risk and opportunity resolution and the second is schedule compression. It has increased the total number of cost drivers from 14 to 16, though this research is continuing [19].

2.8 COCOMO-II

Initially, COCOMO-II was published in Annals of Software Engineering in 1995 [20]. It has three sub-models for estimation of software projects: Applications Composition sub-model, early design sub-model, and post-architecture sub-model.

These projects are simple enough to be rapidly developed from components like applications composition sub-model suitable for computerized aided software engineering (CASE) tools for rapid application development (RAD). Modern GUI builders, and database managers, etc. The applications composition model is based on new Object Points [21].

Early design sub-model is a high-level model that is used to get estimates of the cost and duration of a project, using several new cost factors and new estimate formulas. This sub-model is applicable before the entire architecture of the project is determined. Based on the set of unadjusted functional points or KSLOC, seven effort multiplier (EM) and five scale factors (SF) when available. Early design model requires cost estimation at a general level.

Post-building sub-model is used after the entire construction project has been developed and established. It is closest to intermediate COCOMO '81. It is based on a set of SLOC and FP and 5 SF and 17 EM. The post architecture model provides more accurate cost estimates. COCOMO II's man-hour estimation model is summarized in equation 13 used for both early design and post architecture model for man-hour estimation.

$$\mathrm{PM} = \mathrm{A} \times \mathrm{Size}^{\mathrm{E}} \times \prod_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{EM}_{\mathrm{i}} \quad (13)$$

In Equation (13), *PM* is the effort in person-months, the inputs are a constant, A, the Size of software, an exponent, *E*, and a number of effort multipliers (EM) which depends on the model.

3. EXPERTISE-BASED/CONSENSUS-BASED TECHNIQUES

Expert estimation techniques are based on the ability of one or more people, called experts in software development, to work to estimate software development efforts. Expertise-based techniques are useful if there are restrictions on retrieving quantified empirical data or requirements collection. The various techniques are the following:

3.1 Delphi Technique

Delphi's method [22] is a famous expert method developed in the late 1940s as a prediction method for predicting future events. It is used to guide groups of people to a consensus on certain issues by combining opinions from experts and by preventing bias. In Delphi's method, special meetings are held among project specialists to obtain true information. The procedure is as follows.

- Participants will be asked to receive the quote from the coordinator and individually evaluate specific issues.
- Each expert presents his / her estimate evaluation without consulting with other participants participating in the exercise.
- The coordinator collects all forms. The results of the first round are summarized in a table, and then the form is returned to each participant for the second round.
- In the second round, the participant will be informed of what other participants did in the last round and will be asked again to evaluate the same issue.
- These steps are repeated until approval is obtained for the concerned problem.

3.2 Wideband Delphi Technique

The Delphi methods are improved, and its name is changed into Wideband Delphi. Compared to the original Delphi way to avoid interaction and communication between participants, the Wideband Delphi method includes a group discussion between evaluation rounds. This is a consensus-based approach that gains agreement on the estimate of effort by a group of experts and functions [23].

- Product specifications and estimate forms are distributed to experts by moderators.
- Group meetings are invoked by moderators, among which experts discuss probable problems.
- Each expert fills out an estimate anonymously.
- The moderator gathers estimates, summarizes them, and distributes a summary of estimates.
- Another meeting will be held focusing on discussing the points where the expert estimates are largely different.
- The expert anonymously fills out the quote form and the moderator gathers estimates and summarizes them. This process is repeated until convergence to the estimate is achieved.

3.3 Work Breakdown Structure

Work breakdown structure (WBS) is a way to arrange project elements in a hierarchy. This is based on decomposing the work performed by the project team into smaller subsystems to identify the individual tasks. The software WBS method generates two kinds of hierarchies. One represents a software product and the other represents the activities necessary to build the product. The product hierarchy shows the basic structure of the software, i.e., how different software components characterize the entire system and the activity hierarchy specifies the activities associated with a software component.

For each task, an estimate is made on the amount of work required to complete the task. If the probability is assigned to the cost associated with each element of the hierarchy, the overall estimate of the system can be achieved from the bottom up process of the total development cost of the project. A general algorithm for generating WBS [24]. As project WBS changes over time as requirements and constraints change, it is necessary to avoid confusion using configuration management techniques [25].

3.4 Rule-Based Systems

Rule-based systems utilize knowledge of human experts to solve those real-world problems which require human brainpower. Rule-based systems are built around a set of rules that exist in working memory and are activated by facts that activate and assert new facts. Allow one rule to trigger another rule, so chaining is done in this way [26].

Instead of representing knowledge in a static way, a group of things that are true, the rule-based system uses the IF-THEN statement. In a rule-based system, expertise is represented by a set of rules that describe what to do or what can be concluded in a situation. **Fig. 1** shows the general structure of Rule-based Systems.

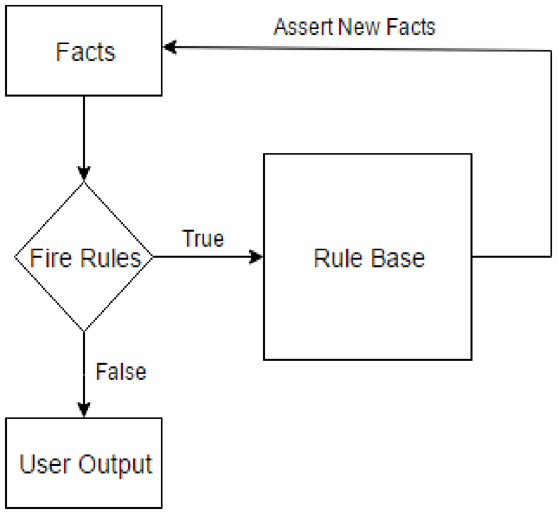

FIG.1. GENERAL STRUCTURE OF RULE-BASED SYSTEMS

3.5 Planning Poker

Planning poker is an expert judgment-based effort estimation method which was first defined by James Grenning in 2002. This technique creates an effort estimate by combining the opinion of multiple experts [27]. Members of planning poker are all the developers on the team i.e. programmers, analysts, testers, product owner, etc., given that the product owner only participates in the process but does not make an estimate.

Firstly, a user story is presented in a planning poker session, and if the story needs more explanation, it is being discussed. A deck of cards is given to each team member with one of the valid estimates written on every card, and then each estimator privately selects a card from those available cards that represent his estimate. All the team members turn over their card simultaneously and everyone can see each estimate.

To be in the ideal state, if the same estimate is selected by everyone then that will be chosen as the official estimate. If estimates differ significantly, members explain and discuss their presented estimates. After the discussion, another round is played. If the estimates do not converge by the second round, the process is repeated until consensus is reached. To be in the worst case, if no consensus is achieved, the story can be deferred for later estimation.

3.6 Top-Down Approach

The design of the Top-Down approach was promoted by IBM researchers Harlan Mills and Niklaus Wirth in 1970s. In the top-down estimation approach, the project total cost estimate is derived from the global property of the software product, using either algorithmic or non-algorithmic approaches. After that the project is partitioned into different components and subcomponents; that is why, in some cases, this approach is also known as a synonym of decomposition. Once a total cost of the project is estimated, a proportion of that cost is assigned to each component. To estimate the project cost accurately, the top-down estimating method requires a history and knowledge of project pricing.

3.7 Bottom-Up Approach

In the Bottom-Up estimating approach, the cost estimation process starts with the lowest level components of the software system. The cost of each component is estimated separately by the people who will be responsible for developing the component. After that these individual estimated costs are aggregated and rolled up to the highest level to determine an estimate for the overall software product.

The prerequisite for this approach is that an initial design of the system must be in place which clearly specifies the decomposition of system components and the work is generally represented by a Work Breakdown Structure. This approach looks at the costs from a more granular viewpoint that is why the estimates are normally more accurate than the other methods.

4. LEARNING-ORIENTED TECHNIQUES

Learning-oriented technology uses prior knowledge as well as current information to develop software cost, estimation models. These techniques learn from previous experiences and build a model to automate the estimation process.

4.1 Case-Based Reasoning

The CBR (Case-based reasoning) model assumes that similar cases have similar solutions. The system continuously learns without depending on experts, the learning process accumulates the resolved case (solution) in the database and makes it accessible to solve new problems in the future [28]. Candidate issues are resolved by finding similar cases from databases containing past projects and applying solutions to finding cases to it. The CBR method is described with the following description.

- Retrieve the most similar case(s) from the memory which is composed of a problem, its solution, and, normally, footnotes regarding how the solution was achieved.
- Reuse the knowledge contained in the retrieved case to solve the target problem and adapt the solution as per the need to fit the new situation.
- Revise the new solution if necessary, after testing it in the real world or a simulation.
- Retain the resultant experience as a new case in the memory, which can be used later while solving any other new problem(s) in the future.

The CBR system includes a preprocessor that organizes the input data for processing, a similarity function to search for similar cases, a predictor to generate a prediction, a predictor for estimating the output value of the subject case, and an update and has a memory updater [29].

4.2 Neural Networks

Neural networks (NN) traces its origins to Warren McCulloch and Walter Pitts' research which created a neural networks calculation model [30]. These estimation models are trained using historical data and automatically adjust their algorithm parameter values to reduce prediction error. Most of the models developed using NN often use a back-propagation training feedforward network called a backpropagation network. In the context of software estimates, backpropagation networks are the most common form of neural networks. To develop a neural model, follow the following steps.

- Define the number of layers in the neuron.
- Define the number of neurons in each layer.
- Determine how all of them are connected.
- Determine the weighted estimation function between the nodes.
- Determine the specific training algorithm to use.
- Once the network is built, the model is trained by giving a series of historical project data as the input project and the corresponding actual cost value.
- The model repeats its learning algorithm and the parameter values of its estimation function are adjusted automatically.
- The estimate of the model and the actual cost/schedule value must be specified to prevent the model from being over-trained theoretically until the iteration is in a predefined delta I will.
- When the training is complete and the proper weight of the arc of the neural network is determined, a new input is provided to the network to predict the corresponding estimate.

4.3 Genetic Algorithm

The genetic algorithm (GA) was invented by John Henry Holland in the 1970s based on the theory of "survival of fittest". The general elements of GA are chromosome populations, selection by fitness, crosses to make new descendants, and random mutations of new offspring [31]. The solutions to the candidate problem are represented by fixed-length binary strings called chromosomes. It is also possible to measure the suitability of any solution and a more appropriate solution is closer to the optimal solution.

Different manipulations, such as selection, crossing, and mutation, are performed on those selected chromosomes when a selection of chromosomes with higher fitness values is made. As a result, a new population is generated, and the process leads to an optimal solution. Due to some potential fluctuations, the basic process of GA is as follows.

- Generate a population of chromosomes, i.e. a group of solutions randomly.
- Apply the genetic operator to the most suitable chromosome or the most suitable chromosome pair to generate a new population from the previous chromosome.
- Step 2 is repeated until the best solution compatibility is satisfied or a specific generation number is generated.

The best solution of the previous generation is considered as an optimal best approximation for a given problem.

4.4 Genetic Programming

Genetic programming (GP) is an extension of GA and does not limit chromosomes to fixed-length binary strings. The first statement of the modern GP based on tree structure was given by Nichael L. Cramer in 1985 [32], after which this study was greatly extended by John R. Koza, an important supporter of genetic programming. In GP, chromosomes are programs that are run to obtain the necessary results. All solutions are easily evaluable algebraic expressions. Crossover, reproductive, and mutation are part of the genetic operation. The crossing operator randomly selects a node from the first chromosome called intersection 1 and the branch to that selected node is disconnected. Next, another node called intersection 2 is randomly chosen from the second chromosome and the branch to that selected node is broken. Thereafter, the two subtrees generated under these cuts are exchanged, and this operation creates a new child. The regeneration operation is a replication of the top n percent of the solution from one generation to the next generation of the genetic algorithm chromosome population as measured by fitness. In a mutation, clauses chosen from chromosomes are altered to maintain genetic diversity from one generation to the next.

5. DYNAMICS-BASED TECHNIQUES

Compared with many other techniques, dynamics-based techniques consider effort and cost factors to be inherently dynamic as they vary over the period of the system development process. Changes in factors such as design requirements, budget, due date, project time training needs, etc. will change the productivity of project personnel.

5.1 System Dynamics Approach

The system dynamic approach was devised to analyze and understand the dynamic behavior of complex systems by Massachusetts Institute of Technology's Jay Forrester in 1961. This is a simulation modeling methodology that displays results and behavior as a graph of information that varies over time. In the system dynamics approach, the model is represented as a modified network with positive and negative feedback loops. The system dynamics simulation model is represented by the set of first-order differential equations [33]as shown in Equation (14).

$$x'(t) = f(x, p) \tag{14}$$

In Equation (14), x is a vector that describes states in the model, t is time, f is a vector function, which is nonlinear, and p is a set of model parameters.

Within the past decade, system dynamics has been successfully implemented for software engineering estimation models [34]. It is difficult to coordinate these technologies, but it is suitable for planning and management.

6. REGRESSION-BASED TECHNIQUES

Regression-based methods are very popular in model building and are used in combination with model-based methods. They provide mathematical algorithms that estimate software costs as a function of key cost factors.

6.1 Standard Regression

Standard regression is based on the ordinary least squares method (OLS) using the linear regression model for estimating unknown parameters. This technique is popular because it is simple and easily accessible from many statistical software packages. The model using the OLS method can be written as in Equation (15).

$$y_t = \beta_1 + \beta_2 x_{t2} + \cdots + \beta_k x_{tk} + e_t \qquad (15)$$

In Equation (15), y_t represents the response variable for the t^{th} observation, β_1 depicts an intercept parameter, $\beta_2 \ldots \beta_k$ are response coefficients, $x_{t2} \ldots x_{tk}$ are predictor (or regressor) variables for the t^{th} observation, and the error term, e_t is a random variable with a probability distribution. In general, the software estimation model is evaluated by error, and this method minimizes the sum of the squared absolute defect, so the problem of this method does not match the evaluation criteria.

6.2 Robust Regression

Robust regression is an enhancement of the OLS approach and problems with outliers in observed software engineering data are mitigated by this approach. Due to the definition of software metrics, various software development processes, and lack of agreement on the availability of qualitative data and quantitative data, there are many outliers in the software development dataset. Robust regression has been applied in [35] to screen outliers for software metric models.

Many parametric cost estimation models have adopted some form of regression-based approach due to their simplicity and being widely accepted. The problem with this technique is that you can eliminate outliers without direct reasoning.

6.3 Fuzzy Logic-Based Methods

The term fuzzy logic was introduced in 1965 of fuzzy set theory [36]. In contrast to Boolean logic where the truth value of a variable is an integer value of 0 or 1, fuzzy logic deals with the concept of partial truth. That is, the truth value of a variable is any real number between 0 and 1.

6.4 Fuzzy Systems

A fuzzy system is a mapping between semantic terms, for instance, "huge", connected to factors. The input and output of the fuzzy framework are either numerical or etymological. A fuzzy framework has three center parts: enrollment works, a standard base, and a yield joining capacity. Fuzzy frameworks have been utilized and for programming improvement models [37]. The upside of fuzzy frameworks is that a very instinctive model can be delivered by utilizing semantic mappings which can be comprehended by anybody without the requirement for any preparation. The impediment of fuzzy frameworks is the trouble in indicating a framework with high precision while keeping up a dimension of significance. More exactness requires more principles, more guidelines lead to progressively complex frameworks, and increasingly complex frameworks are less interpretable.

7. SIZE-BASED ESTIMATION TECHNIQUES

In this section, those software estimation techniques are discussed which are used to predict the software size for software development projects.

7.1 Function Points

Function point (FPs) were presented by Allan Albrecht in 1979 which gauge work hours by assessing the number of capacities [38]. This strategy processes a practical size estimation of programming and the expense is determined from past activities. FPs are increasingly appropriate for the MIS space however risky in the continuous programming area. FPs are not relative qualities, rather they are supreme qualities. Totally a complete number of capacity focuses, it is obligatory to check client capacities, which are of following five sorts. External input (EI) type is information or control client input types and the information crosses the limit from outside to inside in this procedure. External yield (EO) type is the yield information types to the client and the determined information goes over the limit from inside to outside in this procedure. External Inquiry (EQ) type is intelligent data sources, which require a reaction. Internal sensible record (ILF) type is a gathering of consistently related information, which is utilized and shared inside the framework limit. External interface document (EIF) type is a gathering of consistently related information, which is utilized for reference reason as it were. The EIF is an ILF for another application. ILF and EIF are data function types and EI, EO, and EQ are transactional function types. To determine function points, Albrecht uses the average weights through Equation (16).

$$\mathrm{FPs} = (\mathrm{Inp} \times 4) + (\mathrm{Out} \times 5) + (\mathrm{Eq} \times 4) + (\mathrm{MF} \times 10) \qquad (16)$$

In Equation (16), Inp is a number of inputs, *Out* depicts a number of outputs, *Equation* represents the number of inquiries, and MF is for master files (interfaces).

7.2 Full Function Points

Full Function Points (FFPs) measure is especially custom fitted to constant and installed programming spaces. FFP is an expansion of the standard Function Point Analysis (FPA) strategy. It presents two extra control information work types and four new control value-based capacity types [39].

7.3 Use Case Points

Use Case Points (UCPs) system was created by Gustav Karner in 1993 and it depends on comparative standards as the FP estimation procedure. It was intended for the specific arranged frameworks and framework prerequisites being composed utilizing use cases, which is a piece of the Unified Modeling Language (UML) systems. Based on components of the framework use cases, the UCP is determined to quantify the product measure, which is then used to evaluate the task exertion. The UCP condition is comprised of three factors, for example, Unadjusted Use Case Points (UUCP), Technical Complexity Factor (TCF), and Environment Complexity Factor (ECF). On the off chance that the Productivity Factor (PF) as a coefficient is incorporated into it, the condition in half can be utilized to evaluate the number of worker hours required to finish a task [40] in Equation (17).

$$\mathrm{UCP} = \mathrm{UUCP} \times \mathrm{TCF} \times \mathrm{ECF} \times \mathrm{PF} \qquad (17)$$

8. COMPOSITE TECHNIQUES

It can be said vide Table 3, Table 4, and Table 5 that none of the techniques is perfect for every situation; rather they have their own merits and demerits. Composite methods settle this issue as they incorporate at least two procedures to define the most reasonable useful frame for estimation through which the cons of a strategy can be concealed by the geniuses of an alternate one.

8.1 Bayesian Approach

Bayesian examination [41] is a keen evaluating approach which was utilized for the advancement of COCOMO II show. This composite strategy joins skill based and demonstrate based (COCOMO II) procedures. The Bayesian methodology has every one of the benefits of "Standard" relapse, however, it incorporates earlier learning of specialists. It enables the examiner to utilize both example (verifiable task information) and earlier (master judgment) data which is changed to post-information or back perspectives. The two data sources can be joined utilizing Bayes' hypothesis as pursues in Equation (18).

$$f(\beta/Y) = \frac{f(Y/\beta)f(\beta)}{f(Y)} \qquad (18)$$

In condition Equation (18), β is the parameter of the vector about which concerned, Y speaks to the vector of test perceptions from the joint thickness work f (β/Y), f (β/Y) means the back thickness work for β which outlines all the data about β, f(Y/β) speaks to the example data, and f (β) is for the earlier data that abridges the master judgment data about β.

9. OTHER TECHNIQUES

Apart from the estimation techniques discussed above, many techniques have been proposed to date because the research is going on in this hot field. All of them cannot be covered to the fullest, though some other techniques are briefly discussed in this section.

9.1 Cost-to-Win

Cost to-win is a method in which the expense of programming is evaluated to be the best cost to win the task. Rather than the product usefulness, the fundamental focal point of this estimation is the client's financial plan [42]. For example, if the client can bear the cost of 80 man-months, however the sensible cost estimation for a venture is 120 man-months, at that point by utilizing the cost to-win procedure, by and large, the estimator is requested to alter the exertion estimation to fit 80-man-months to win the task. The upside of this system is that you win the agreement. The hindrance of this training is that possibly it causes a terrible postponement in conveyance or powers the improvement group to stay at work past 40 hours for example time and cash run out before the activity is finished.

9.2 Parkinson

Utilizing Parkinson's Law "work grows to fill the accessible volume" [43], the task cost is controlled by whatever assets are accessible rather than target evaluation. For instance, if a venture must be conveyed in 10 months and 5 individuals are accessible to chip away at it, at that point the exertion is assessed to be 50 men per months. The advantage of this technique is that it provides a good estimation and there is no overspend on the project. The disadvantage of this method is that it gives unrealistic estimates and does not promote good software engineering practices.

9.3 PROXY-Based Estimating

PROXY-based estimating (PROBE) for personal software process (PSP) was introduced by Watts Humphrey[44] to estimate size and effort. PROBE is based on the idea that similar projects will take about the same effort. PROBE utilized the KLOC with LOC to measure the size of effort but can be easily customized for FPs, or other levels of granularity as per need. The advantage of this process is that it works better at the individual small effort level. The limitation is that it does not scale well to larger efforts.

9.4 Delphi Technique

Delphi technique effort technique that involves the experts from the estimation. The team consists of the 4-8 members with the moderator [45]. The estimation process starts from kickoff meeting, creates wbs, discussion about the list of assumptions, effort estimation for each task, achieve consensus. The Delphi model is the Looping process which is used to filter the judgments of experts by using a series of questionnaires.

9.10 Effort Estimation using the Machine Learning Algorithm

The approach for the deploying and duration of the effort. Three algorithms of machine learning SVM, MLP and GLM are used with cross-validation [46]. Their results indicate the good accuracy and suitability for the deployment.

9.11 Cost Estimation using an Artificial Neural Network

Cost estimation is the most challenging task for software project management. The results are generated from the multilayer neural system [46]. They create the quantitate measure in the proposed model.

9.12 Hybrid Model of Input Selection Procedure

Software effort estimation is the predicting of development effort and development time required to develop any software project. It is the main step of the software development process and at the same time measured to be the key task as correct calculations of growing of the current project. Software cost estimation is done by the input selection procedure to find the cost drivers relevant leave the irrelevant attributes [47].

10. DISCUSSION

In this section, all the techniques discussed above are summarized in tabular form. Tables 3-6 overview the learning-oriented techniques, parametric/ algorithmic models, the expertise-based/ consensus-based techniques, and size-based estimation techniques for ready reference, respectively.

The tables in this section clearly describe the several techniques advantages and disadvantages. Some of the drawbacks are insignificant and can be tolerated, but some of the pitfalls are crucial enough and cannot be ignored.

11. CONCLUSION

This study discusses various software development cost estimation models and techniques along with their pros and cons. Software development effort and cost estimation is an interesting area in software development. The reliable effort estimation technique which provides accurate estimates for a software engineer in software development is needed. The study tells that none of the techniques is best for all situations in software development; rather they are applicable in different nature of projects and are challenged by the rapidly changing software industry. Since no single technique gives a hundred percent accuracy, that is why one technique is not chosen.

12. RECOMMENDATIONS

In our opinion, it is recommended to use a hybrid model for estimating the cost of software project development. Through a hybrid approach, one technique will complement the other; and already built famous models can be improved in this way. Secondly, we should calibrate those models which provide calibration support because if a model was developed in a different environment, we cannot expect it to perform very well in a new environment, so recalibration helps in such situations. The achievement of this study is that various techniques are discussed in this study and it helps in software cost estimation during the development of software.

E 3. SUMMARY OF LEARNING-ORIENTED TECHNIQUES

Methods	Proposed by	Year	Strengths	Limitations
CBR	Roger Schank and his students at Yale University	The 1980s	• Simplified procedure for knowledge acquisition. • Improves eventually as case base grows • The learning ability of CBR systems helps in maintaining knowledge • High user acceptance.	• Does not tackle categorical / nominal data. • Intolerant of irrelevant features and noise. • Large storage space can be taken for all cases. • Adaptation is often required for retrieved cases which may be quite difficult or impossible in many domains.
Neural Network	Warren McCulloch and Walter Pitts	1943	• Can solve complex problems • Requires less formal statistical training. • Can detect any complex relationships between dependent and independent variables.	• Difficult to design • Needs training to operate. • High processing time is required for large NNs. • Too much of a black box nature.
GA	John Henry Holland	The 1970s	• Easy to understand concepts • More chances of getting the optimal solution • Intrinsically parallel • Does not depend on specific knowledge of the problem.	• Chromosomes representing individuals have to be a fixed length binary string. • Not a silver bullet to solve a problem. • Low usability if the algorithm is not trained long enough. • More complex to implement.
GP	Nichael L. Cramer	1985	• Eliminates the constraint that the chromosomes representing individuals have to be a fixed length binary string. • Provides better accuracy. • Every solution can be evaluated as it is an algebraic expression. • Little specific domain knowledge is required.	• More exertion in setting up and preparing is required. • The inescapable tradeoff between exactness from unpredictability and simplicity of elucidation.

TABLE 4 . SUMMARY OF PARAMETRIC / ALGORITHMIC MODELS

Methods	Proposed by / Owner	Year	Size Input	Estimates What	Activities covered (MBASE/RUP or ISO/IEC 15288 phases)	Limitations	Tools (if any) / Cost
SLIM	L. H. Putnam	1978	SLOC	Effort, Time	Inception, Elaboration, Construction, Transition and maintenance	Uncertainty about LOC (software size) in the early stages may lead to inaccurate estimates. Not suitable for small projects. Only considers time and size, not all other aspects of SDLC.	A SLIM suite of tools / Commercial
COCOMO	B. W. Boehm	1981	KDSI	The effort, Cost, Time	Plan and requirement, preliminary design, detailed design, code, Integration & Testing	Hard to accurately estimate KDSI early on in the project. Achievement depends to a great extent on adjustment utilizing recorded information which isn't constantly accessible. Unsuitable for large projects as much data is required. Vulnerable to misclassification of development mode. Lack of factors root limited accuracy. Assumes the requirements to be stable and predefined.	Costar 7.0 / Commercial
SEER-SEM	Galorath Inc.	1983 and The 1990s	SLOC, FPs, UCs	Effort, Cost, Risk, Duration	Inception, Elaboration, Construction, Transition and maintenance	It takes many parameters as input which increases the complexity and uncertainty. The exact size of the project is a key concern in this model.	SEER for Software / Commercial
Checkpoint	Capers Jones	1997	FPs	Effort, Cost, Schedule, Defect	Inception, Elaboration, Construction, Transition and maintenance	Estimation is bit complex since it is done at activity-level and task-level.	Checkpoint/ Commercial
ESTIMACS	Howard Rubin	The 1970s	Function-Point-like	Effort, Cost, Risk	Inception, Elaboration, Construction	Each stage does not clearly translate the effort. The results of the package are not totally explainable.	Estimacs / Commercial
PRICE-S	RCA	1970s	SLOC, FPs, POPs, UCCP	Cost, Schedule	Inception, Elaboration, Construction, Transition and maintenance	Model is presented as a black box to the users because its core concepts and ideas are not publicly defined.	TruePlanning/ Commercial
COSYSMO	Ricardo Valerdi	2002	Requirement, Interfaces, Algorithms, Operational, Scenarios	Effort, Time	Conceptualize, Develop, Activity, Test, and Evaluation, Change to Operation, Work Maintain or Enhance, and Supplant	It overlaps with the COCOMO II model causing needless double-counting of effort; as in most organizations, software engineering and systems engineering are highly coupled.	SystemStar / Commercial
COCOMO II	B. W. Boehm *et al.*	Research started in 1994, published in 1995	KSLOC, FPs, Application Points	Cost, Effort, Schedule	Inception, Elaboration, Construction, Transition and maintenance	Its 'heart' is still based on a waterfall process model. Duration calculation for small projects is not reasonable.	USC COCOMO II / Free

TABLE 5 . SUMMARY OF EXPERTISE-BASED / CONSENSUS-BASED TECHNIQUES

Methods	Proposed by	Year	Strengths	Limitations
Delphi	Olaf Helmer *et al.*	The 1940s	• Simple to manage • Easy to use • Quick to derive an estimate • Useful when in-house experts of the organization cannot come out with a quick estimate • Results can be much accurate if experts are chosen carefully	• Avoid group discussions • Too simplistic • Hard to locate appropriate experts • The derived estimate is not verifiable
Wideband Delphi	B. W. Boehm	1981	• Supports group discussion among assessment rounds	• Time is needed and several experts take part in the process.
WBS	The concept developed by US DoD	Developed by the US Navy in 1957. Published in June 1962 by DoD, NASA, & aerospace industry.	• Good for planning • Good for control • Has detailed steps	• Development of WBS is not so easy • Step by step approach is a heck of a job • Difficult to find the most accurate level of details
Rule-based Systems	AI researchers	?	• Uses IF-THEN statements and does not follow a static way to represent knowledge • Simplicity - the natural format of rules • Uniformity – the same structure of all rules	• If not specially crafted, infinite loops can occur • The computational cost can be very high as rules require pattern matching • Rules cannot modify themselves
Planning Poker	James Grenning	2002	• Enjoyable method for estimation • No first-estimate bias because of the confidential individual estimate • Discussion leads to better estimates	• Time-consuming • Export-dependent • Less accurate results if the team have no prior experience with similar tasks
Top-Down	IBM researcher Harlan Mills and Niklaus Wirth	Promoted in the 1970s	• System-level focus - captures system-level effort like component integration, users' manual, and change management • Requires minimum project details • Easier to manipulate	• Lacks a thorough breakdown of sub-components • Does not discover tricky low-level technical problems which are liable to increase costs • Provide little detail on cost justification
Bottom-Up	?	?	• Sustains project tracking • Based on an exhaustive analysis • Transparency – potential errors can be investigated plus their impact can be tested because of the detailed cost data	• Much estimation effort is needed. • Chances of over-estimate are there as each level folds in another level • Hard to make estimates early in the lifecycle

A question mark (?) shows that authors were not capable to find information from the related work.

TABLE 6. SUMMARY OF SIZE-BASED ESTIMATION TECHNIQUES

Methods	Proposed by	Year	Strengths	Limitations
FP	Allan Albrecht	1979	• Make estimation possible early in the project lifecycle. • Independent of how the requirements of the software were expressed. • Does not depend on a specific technology or programming language.	• Not capable of dealing with hybrid systems. • No availability of enough research data as compared to LOC. • Time-consuming method.
FFP	Denis St-Pierre et al.	1997	• Can cope with real-time software domain. • Can cope with embedded software. • Retains the actual FPA quality characteristics.	• Restricted range of software (specifications) that can be sized is covered. • Time-consuming method.
UCP	Gustav Karner	1993	• The process can be automated. • Can be measured early in the project lifecycle. • Easy to use. • When estimation is performed by skilled people, estimates would be close to the actuals.	• Only applicable for those software projects whose specification can be expressed by use cases. • UCP is less useful in iteration tasks in the team.

ACKNOWLEDGMENT

This research work is supported by the Air University Islamabad, Kamra Campus, Pakistan, COMSATS University Islamabad, Wah Campus, Pakistan, National Textile University, Faisalabad, Pakistan, Prince Sultan University, Riyadh, Saudi Arabia and University of Engineering and Technology Taxila, Pakistan. The authors are very thankful to all these

universities for providing the required resources to perform this research.

REFERENCES

[1] Mendes, E., Watson, I., Triggs, C., Mosley, N., & Counsell, S., "A comparative study of cost estimation models for web hypermedia applications", Empirical Software Engineering, Vol. 8, No. 2,pp 163-196, 2003

[2] Rashid, J. and Nisar, M.W., "How to Improve a Software Quality Assurance In Software Development-A Survey". International Journal of Computer Science and Information Security, Vol. 14, No 8, pp.99-108, 2016.

[3] Musawwer, K, Rashid, J. and Nisar, M.W., "A CMMI Complaint Requirement Development Life Cycle", International Journal of Computer Science and Information Security, Vol. 14, No 9,pp. 1000-1009, 2016.

[4] Tamrakar, R. and Jørgensen, M., "Does the use of Fibonacci numbers in Planning Poker affect effort estimates?", 16th International Conference on Evaluation\& Assessment in Software Engineering, pp. 228-232, 2012.

[5] Kumar, S., Rastogi, R. and Nag, R.,"Limitations of Function Point Analysis in Multimedia Software/Application Estimation", In Software Engineering, Springer, pp. 383-392, 2019.

[6] Wen, J., Li, S., Lin, Z., Hu, Y. and Huang, C., "Systematic literature review of machine learning based software development effort estimation models", Information and Software Technology, Vol. 54, No 1, pp.41-59, 2012.

[7] Hani, S.U., Alam, A.T. and Shaikh, A.B., "Tuning COCOMO-II for software process improvement: A tool based approach", Mehran University Research Journal Of Engineering & Technology, Vol. 35, No 4, pp.505-522, 2016.

[8] Rashid, J., Mehmood, W. and Nisar, M.W., "A Survey of Model Comparison Strategies and Techniques in Model Driven Engineering", International Journal of Software Engineering and Technology, Vol. 1,No 3, pp.165-176, 2016.

[9] Putnam, L.H., "A general empirical solution to the macro software sizing and estimating problem", IEEE transactions on Software Engineering, Vol. 4, No 4, pp.345-361,1978

[10] Boehm, B.W., "Software engineering economics ", IEEE Transations on Software Engineering,Vol. 10, No 1, pp.4-21, 1984.

[11] Kemerer, C.F.," An empirical validation of software cost estimation models", Communications of the ACM, Vol. 30, No 5, pp.416-429,1987.

[12] Leung, H. and Fan, Z., "Software cost estimation", In Handbook of Software Engineering and Knowledge Engineering, Vol. 2, pp. 307-324, 2002.

[13] Fischman, L., McRitchie, K. and Galorath,"Inside seer-sem", CrossTalk, p.146,2005.

[14] Capers, J.," Applied software measurement", McGraw-Hill, 1996

[15] Heemstra, F.J., "Software cost estimation", Information and software technology, Vol. 34, No 10, pp.627-639, 1992.

[16] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. and Selby, R., "Cost models for future software life cycle processes: COCOMO 2.0", Annals of software engineering, Vol. 1, No 1, pp.57-94,1995.

[17] Valerdi, R. and Boehm, B.W., "COSYSMO: A systems engineering cost model",2010

[18] Valerdi, R., Boehm, B.W. and Reifer, D.J., "COSYSMO: A Constructive Systems Engineering Cost Model Coming of Age", In INCOSE International Symposium , Vol. 13, No 1, pp. 70-82,2003.

[19] Wang, G., Boehm, B., Valerdi, R. and Shernoff, A., 2008,"Proposed Modification to COSYSMO Estimating Relationship", In INCOSE International Symposium, Vol. 18, No. 1, pp. 249-262,2008.

[20] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. and Selby, R., "Cost models for future software life cycle processes: COCOMO 2.0", Annals of

software engineering,Vol. 1, No 1, pp.57-94,1995.

[21] Banker, R.D., Kauffman, R.J. and Kumar, R., "An empirical test of object-based output measurement metrics in a computer aided software engineering (CASE) environment", Journal of Management Information Systems,Vol. 8, No 3, pp.127-150,1991.

[22] Dalkey, N., "An experimental study of group opinion: the Delphi method", Futures, Vol. 1, No 5, pp.408-426,1969.

[23] Clark, B.K.,"The effects of software process maturity on software development effort",1997.

[24] Tausworthe, R.C.,"The work breakdown structure in software project management", Journal of Systems and Software, Vol. 1, pp.181-186,1979

[25] Globerson,S.,"Impact of various work-breakdown structures on project conceptualization", International Journal of Project Management, Vol. 12, No 3, pp.165-171,1994.

[26] Gray, A. and MacDonell, S.G., "A comparison of techniques for developing predictive models of software metrics",1997.

[27] Molokken-Ostvold, K. and Haugen, N.C., "Combining estimates with planning poker--an empirical study", ASWEC, pp. 349-358, 2007.

[28] Zima, K., "The Case-Based Reasoning model of cost estimation at the preliminary stage of a construction project", Procedia Engineering, Volume 122, pp.57-64,2015.

[29] Aha, D.W., "Case-based learning algorithms", DARPA Case-Based Reasoning Workshop , Vol. 1, pp. 147-158,1991.

[30] McCulloch, W.S. and Pitts, W., "A logical calculus of the ideas immanent in nervous activity", Bulletin of mathematical biology, Vol. 52, No 1, pp.99-115,1990.

[31] Mitchell, M., "An introduction to genetic algorithms", MIT press,1998.

[32] Cramer, N.L.,"A representation for the adaptive generation of simple sequential programs", In Proceedings of the first international conference on genetic algorithms , pp. 183-187,1985.

[33] Madachy, R.J. and Khoshnevis, B.,"A software project dynamics model for process cost, schedule and risk assessment",1994.

[34] Abdel-Hamid, T.K., "The dynamics of software project staffing: a system dynamics based simulation approach", IEEE Transactions on Software engineering, Vol. 15, No 2, pp.109-119,1989.

[35] Miyazaki, Y., Terakado, M., Ozaki, K. and Nozaki, H., "Robust regression for developing software estimation models", Journal of Systems and Software, Vol. 27, No 1, pp.3-16,1994.

[36] Yager, R.R., "Connectives and quantifiers in fuzzy sets", Fuzzy sets and systems, Vol. 40, No 1, pp.39-75,1965.

[37] Bastani, F.B., DiMarco, G. and Pasquini, A., "Experimental evaluation of a fuzzy-set based measure of software correctness using program mutation", 15th international conference on Software Engineering, pp. 45-54,1993.

[38] Albrecht, A.J. and Gaffney, J.E., "Software function, source lines of code, and development effort prediction: a software science validation", IEEE transactions on software engineering, Vol. 6, pp.639-648,1983.

[39] St-Pierre, D., Maya, M., Abran, A., Desharnais, J.M. and Bourque, P., "Full function points: Counting practices manual", Software Engineering Management Research Laboratory and Software Engineering Laboratory in Applied Metrics,1997.

[40] Clemmons, R.K., "Project estimation with use case points", The Journal of Defense Software Engineering, Volume 19, No 2, pp.18-22,2006.

[41] Chulani, S., Boehm, B. and Steece, B., "Calibrating software cost models using bayesian analysis", IEEE Transactions on Software Engineering,1999.

[42] Nasir, M., "A survey of software estimation techniques and project planning practices", In Software Engineering, Artificial Intelligence, Networking, and

Parallel/Distributed Computing,pp. 305-310,2006.

[43] Gutierrez, G.J. and Kouvelis, P., "Parkinson's law and its implications for project management", Management Science, Volume 37, No 8, pp.990-1001,1991.

[44] Humphrey, W.S., "A discipline for software engineering", Addison-Wesley Longman Publishing ,1995.

[45] Rai, A., Gupta, G.P. and Kumar, P., "Estimation of software development efforts using improved delphi technique: A novel approach",Int. J. Appl. Eng. Res., Volume 12, No 12, pp.3228-3236,2017.

[46] Pospieszny, P., Czarnacka-Chrobot, B. and Kobylinski, A., " An effective approach for software project effort and duration estimation with machine learning algorithms", Journal of Systems and Software, Volume 137, pp.184-196.2018

[47] Wani, Z.H. and Quadri, S.M.K., "Software Cost Estimation Based on the Hybrid Model of Input Selection Procedure and Artificial Neural Network", Artificial Intelligent Systems and Machine Learning,Volume 10, No 1, pp.18-24, 2018.