Vol: 02 Issue: 02

International Journal of Data Science and Advanced Analytics

Software Cost Estimation: Algorithmic and Non-Algorithmic Approaches

Bilal Khan[1], Wahab Khan[2], Muhammad Arshad[3] and Nazir Jan[4]
[1, 2] City University of Science and Information Technology, Peshawar, Pakistan
[3] Institute of Business and Management Sciences, University of Agriculture, Peshawar, Pakistan
[4] University of Engineering and Technology, Peshawar, Pakistan
[1]bilalsoft63@gmail.com, [2]wahab_suit@yahoo.com, [3]arshad12@aup.edu.pk, [4]engr.nazirjan@gmail.com
Corresponding author email: bilalsoft63@gmail.com

***Abstract*— Software Development Organizations develop a huge number of projects on a yearly basis. One of the main issues is that which tool can be selected for an accurate estimation of the cost of software projects. Software cost estimation (SCE) is one of the main objectives of any project. SCE directly introduces almost all management events including resources allocation, project planning, and project bidding. In this study, some important SCE methods have been studied for comparative analysis and this has been concluded that none of the methods are essential inferior or superior to others, as there is no individual approach that is finest for every situation. The selection of the method is depending upon the nature of the project.**

***Keywords*— Software Cost Estimation, Software Cost Estimation Methods, Algorithmic, and Non-algorithmic Approaches**

1. Introduction

Software projects have become a very affluent element of the computer system in existing years. The huge development cost of software projects is due to human efforts and the utmost SCE methods emphasize on this [1]. Accuracy in the cost estimation is very acute for both the developer and customers. Undervaluing the software costs might result to outstrip the budgets with under-developed functions, low quality and failure to complete the project within time [2]. The size, accuracy and rising complexities of the software projects have excessive effects on the estimation's accuracy. The very important role is of the project management in the supervision of these estimation procedures. A lot of research has been conceded that redirects the increasing loads of high-grade software through operative cost estimation [3]. Many software cost estimation models try to produce an effort estimation, which further can be transformed into the project duration and cost, even though effort and cost are interrelated to each other [4]. There are a number of methods and models which are used for software cost estimation, but it is very difficult to decide which model or method can be selected for cost estimation [4]. In direction to solve such problems, it is very essential to have knowledge about SCE methods and models [5]. To estimate the project cost it is important to recognize and understand the strength and weaknesses of the SCE models to be used [6]. This research gives a comprehensive analysis of each of the SCE methods which could transpire its use in numerous surroundings.

The rest of this paper is organized into 5 sections. Section first starts from the introduction. Section 2 and 3 discuss the software cost estimation techniques and selection mechanism. Section 4 is a conclusion and references are defined in section 5.

2. Software Cost Estimation Techniques

Some of the existing approaches for SCE are listed and summarized in this section. SCE approaches are mainly divided into two main categories. These are algorithmic and non-algorithmic approaches. These approaches utilized the source line of code (SLOC) as input. The first one is discussed in the upcoming section

2.1. Non-algorithmic Techniques

In non-algorithmic models, the estimation can be done by using preceding experience and projects, which is the same as the under-estimation projects. Some of the non-algorithmic approaches are listed here.

2.1.1. Expert Judgment

Expert Judgment (EJ) technique is used widely throughout the generation of cost estimation of software projects. Estimators have to create a large number of suppositions and judgments for predicting the cost of new products. Though the use of EJ is often scowled upon, not well established or implicit by non-cost estimators within a parallel engineering environment [7]. EJ is actually a capability for the prediction of cost estimation of software projects where the procedure used in accessing with groups or personages with expert knowledge or preparations. EJ depends on expert experience, knowledge and motivations, the grand of knowledge on the area and the discussion between analysts and experts. Thus, according to Cooke, the significant tool using EJ is the illustration of hesitation. Ballay expresses the expert that the "person who has the knowledge" and the analyst who continue EJ exercise. However, there is no recognized research on statistics collection methods, there is an extensive sensation that the method must be mainly spontaneous and thus responsible to be individually unfair and delicate to political stresses [8].

Advantages: EJ uses past Experience for cost estimation of the software projects. EJ method is the knowledge from previous

projects that the expert takes to the planned project. EJ methods are appropriate for measuring the variances between previous and imminent programs and are specifically beneficial for new programs for which no past examples occur.
Disadvantages: EJ methods sometimes due to deficient knowledge may produce complications. EJ methods are habitually struggling to precisely estimate the cost of a new software program. They are not repeatedly used unaided in software cost estimation.

2.1.2. Analogy Based Estimation

Analogy-based (AB) cost estimation is a form of Cased Based Reasoning (CBR). Cases are demarcated as notions of events that are partial in space and time [9]. AB cost estimation of software projects is mainly apparent, as it relies on past information from comparable projects, whereby comparisons are resolute by equating the projects' significant features and attributes. Though, one critical side of the AB method is not yet completely accounted for; the dissimilar effect or premium of a project's several features [10]. AB cost estimation of software projects is molded on the moralities of real e orts and values [4]. AB approaches for estimation might mark its use at the system-level and at component-level [5]. In some features, AB is the appropriate forms of EJ as the expert frequently do searching for corresponding conditions and enlightening the sentiments.
Following are the steps for making use of estimation by AB:
1. The planned project remains categorized.
2. Creating the choice of the precise comparable finalized project whose attributes are stocked in an historic database.
In this method, the function of resemblance like Euclidean similarity (ES), Shepperd and Manhattan similarity (MS) are defined which associates the features of two projects.

$$Sim(p,p') = \frac{1}{\sum_{i=1}^{n} \sqrt{Dist\,(fi,f'i) * Wi}} \qquad (1)$$

here p and p' are the projects, fi and $f`i$ shows ith features for each project is 0are001 and exploited for getting the non-zero result, Wi is the weight falls in range from 0 to 1 for n features. MS and ES formulas are somewhat more similar method but it benefits in computing the variance between them [4].
Advantages: In the early phases, AB estimation of projects is a better way when there is very low information available. This method takes less time and is simple and easy to use. Success rates of an organization are probable to be high since the method is grounded on the organization's historical project records. It can be used for the estimation of effort and period of separable responsibilities too.
Disadvantages: Some complications are still antagonized by AB estimation methods, for example, the non-normal features e.g. heteroscedasticity, skewness and extreme outliers of the datasets from software engineering [11] and the growing sizes of the datasets [12]. The non-normal and bulky datasets permanently lead AB estimation approaches for low forecasting accuracy and high computational outlay to relieve these downsides [13].

2.1.3. Bottom-Up and Top-down Approach

It is also known as Macro Model. EJ software development effort may observe bottom-up or top-down approaches. The complete effort estimation may be grounded on possessions of the software project altogether and divided into the project activities (top-down) or premeditated as the sum of the project activity estimates (bottom-up) [14]. The complete SCE is on assessment from the widespread stuff of the software project using a top-down approach for estimation, and the project is distributed in the different subsidiary sections or appliances [5]. Recompense contains events of system-level e.g. documentation, configuration management, project control, integration, etc.
Advantages: This method is generally faster and simple to implement for cost estimation of software projects as it needs the least details of the project. This method is attentive for the system-level activities e.g. documentation, integration, configuration.
Disadvantages: For justifying the estimate or conclusions it does not deliver details. It does not recognize the exertion of low-level problems and can have less correctness that incline to the desertion of lower-level elements and the probability of procedural problems.

2.2. Algorithmic Techniques

Algorithmic models use mathematical equations for cost estimation of software projects. These models estimate the cost based on project type, size, attributes, procedures, and the team involved in the development of software projects. Using algorithmic techniques various models have been established such as FBPA, Putnam's model and COCOMO model [15]. Each of them uses the mathematical equation:

$$Effort = f(x1, x2, x3, \ldots\ldots\ldots, xn) \qquad (2)$$

Here, x1, x2, x3, …., xn are the cost factors.
Some algorithmic models are discussed here.

2.2.1. COCOMO (Constructive Cost Model) Model

Boehm proposed this model which is broadly acknowledged in practice. Using COCOMO effort is measure in person-months and the size of code S is given in a thousand lines of codes (KLOC).

a) *Basic COCOMO*

The basic COCOMO model uses different three sets of {a, b} conditional on the difficulty of the software only, given in Table 1:
This is a simple model and easy to used. The basic COCOMO can only be used as an uneven estimation as several cost factors are not considered.

Table 1 Basic COCOMO Set of (a, b) [16]

Projects	**A**	**a**
Simple	2.4	1.05
Complex	3.0	1.15
Embedded	3.6	1.20

b) *Intermediate COCOMO* and *Detailed COCOMO*

Effort estimation is gained using power function with given three sets of {a, b} in the intermediate COCOMO, in which coefficient is different from that of the basic COCOMO shown in Table 2.

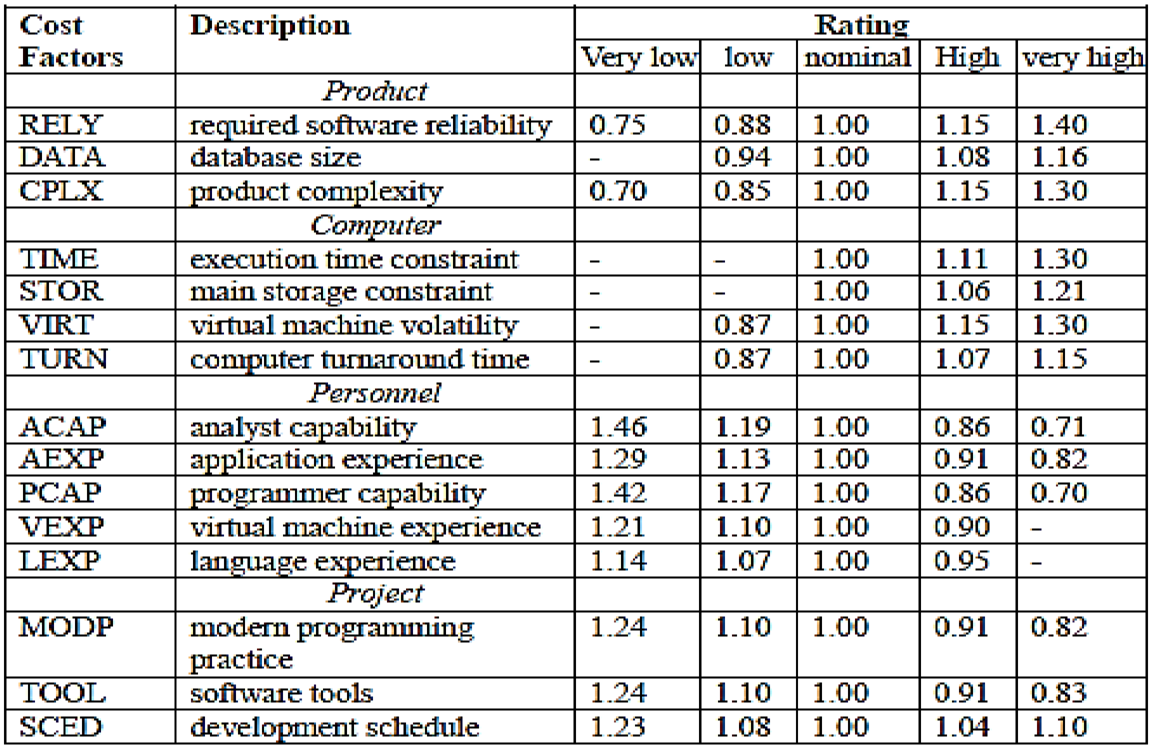
Table 2 Intermediate COCOMO Set of (a, b) [4]

Projects	**A**	**B**
Simple	3.2	1.05
Complex	3.0	1.15

Embedded	2.8	1.20

Then, 15 attributes as cost factors are selected having the values ranging from 0.7 to 1.66 shown in Table 3. The complete impact factor M is achieved as the product of all single factors, the estimate is gained by multiplying M to all nominal estimates. Table 3: The cost factors and their weights in COCOMO II [1].

Table 3 The Cost Factors and their Weights [16]



Cost Factors	Description	Rating				
		Very low	low	nominal	High	very high
	Product					
RELY	required software reliability	0.75	0.88	1.00	1.15	1.40
DATA	database size	-	0.94	1.00	1.08	1.16
CPLX	product complexity	0.70	0.85	1.00	1.15	1.30
	Computer					
TIME	execution time constraint	-	-	1.00	1.11	1.30
STOR	main storage constraint	-	-	1.00	1.06	1.21
VIRT	virtual machine volatility	-	0.87	1.00	1.15	1.30
TURN	computer turnaround time	-	0.87	1.00	1.07	1.15
	Personnel					
ACAP	analyst capability	1.46	1.19	1.00	0.86	0.71
AEXP	application experience	1.29	1.13	1.00	0.91	0.82
PCAP	programmer capability	1.42	1.17	1.00	0.86	0.70
VEXP	virtual machine experience	1.21	1.10	1.00	0.90	-
LEXP	language experience	1.14	1.07	1.00	0.95	-
	Project					
MODP	modern programming practice	1.24	1.10	1.00	0.91	0.82
TOOL	software tools	1.24	1.10	1.00	0.91	0.83
SCED	development schedule	1.23	1.08	1.00	1.04	1.10

Using the basic COCOMO or intermediate COCOMO the software cost estimation in system level, while the detailed COCOMO works on every sub-system individually. The detailed COCOMO is very much suitable for the large systems which contain irrelevant sub-systems.

Advantages: It is easy to understand and implement and have better accuracy. This model uses historical data to work fine, therefore it is very predictable.

Disadvantages: The COCOMO model ignores the documentation, requirements cooperation, knowledge, customer skills, hardware issues, person turnover level, and other parameters. It generalizes the influence of security/safety traits. It depends on the total time consumed in each level.

2.2.2. Agile COCOMO Model

The Agile COCOMO includes the complete algorithmic model for COCOMO. It the most momentous way of cost estimation of software projects based on analogy. It is used to acquire the precise results for the newest projects [4]. Agile COCOMO-II model has been developed by USC-CSE. USC-CSE is a tool for SCE which depend on the COCOMO-II model. Analogy based estimation is used here for the production of accurate output, which is simple and easy to use and learn. We might create estimation for a project in terms of function points, object points, person-months, and the dollar, etc. [6].

2.2.3. Putnam's Model

This model derives based on Rayleigh/Norden's manpower dissemination and analyzing and finding several finalized projects [17]. The essential part of Putnam's model is known as the *software equation* defines as follows:

$$S = E * Efforts^{1/3} * t_d^{4/3} \qquad (3)$$

Here t_d is the delivery time of software, E is the *environmental factor* which replicates the competences of development that can be taken using software equations from historical data. An effort is taken in person year and the size of S is in LOC. Additional essential relation originate by Putnam is

$$Effort = D_0 * t_d^3 \qquad (4)$$

Here D_0 is manpower build-up parameter ranging from 8 (new software) to 27 (remodeled software). The Putnam's model is usually used in preparation and SLIM. SLIM is a tool based on Putnam's model for manpower planning and estimation.

2.2.4. Function Point-Based Analysis

Function Point-Based Analysis (FPBA) is the method of calculating the complexity and size of a software system in the tenure of the functions that the system provides to the user [18]. The provided functions are isolated to the tools or language used for the development of the software project [19]. FPBA is basically planned for the measurement of business type applications, which is not suitable for scientific or technical applications because scientific or technical applications deal with complex algorithms that cannot be handled through FPBA [20]. The FPBA features to overcome the major problems of using Lines of Code (LOC) as a measure of system size. First, function points are autonomous of the tools, language or procedures used for operation; i.e., they do not revenue into concern of processing hardware, database management systems, programming languages, or any other data processing technology [21]. Second, function points can be estimated from design specifications or requirements specifications thus creating it promising to estimate development effort in the initial stages of development. Meanwhile, function points are openly related to the statement of requirements; any variation of requirements can effortlessly be charted by a re-estimation. Third, as function points are grounded on the system user's outdoor opinion of the system, non-technical users of the software system have good consideration of what function points are quantifying [22].

Advantages: FPBA can be used in the initial stage of the software development life cycle. It is autonomous of methodologies, technology, and programming language. FPBA can be used for graphical user interface and can be calculated initially and frequently.

Disadvantages: FPBA depends on individual assessments with the involvement of too many judgments that's why it has low accuracy and very time-consuming. It needs transformation as many cost models and efforts are based on LOC and very less amount of research data is accessible on FPBA equated with LOC.

3. Selection of Estimation Techniques

From the above discussion on the different techniques for cost estimation, this is concluded that there is not any single model that can be recognized as the finest one. The pros and cons of each model are correlated, so a consolidation of these models [23] can help in running out the flaws of any individual method. This can help to increase their individual strength and reduce the adverse effects of the separable model. We also can cross-check one method with another. Typically, it is suggested to use non-algorithmic approaches e.g. expert judgment or estimation by analogy method for the known projects. Instead of less known and larger projects, it verifies that to use algorithmic methods. Among the algorithmic models, COCOMO is much better than the other models. Efforts can be prepared to use an arrangement of the techniques to reach a better estimate of the software.

4. Conclusions

To achieve desired results in term of cost estimation, the software cost estimation can be perceived as a necessary activity that requires the use of both precise methods and techniques. From the assessment of different models, we can accomplish that there is not a particular method of model that is good or bad from one another, actually, their strengths and weaknesses are often complementary to one another. Now the problem as that which method for estimation to be used for particular project estimation? It depends upon the nature of the project. As per the strengths and weaknesses of the individual methods, we can make a choice concerning which method can be selected for the cost estimation of software projects. The project managers are essential to insert values for different drivers as per cost with concerns to data from historical projects. The COCOMO models can deliver all abilities. It produces the cost for new projects in an abundant precise way than several models of cost estimation. Our remarks directed that it can be preeminent to use several cost estimation techniques, and then compare the outcomes, before defining the causes for large variants and documenting any expectations that were made while making the estimates of software projects..

References

[1] H. LEUNG and Z. FAN, "Software Cost Estimation," *Inf. Softw. Technol.*, vol. 34, no. 10, pp. 307–324, 2002.

[2] I. M. Keshta, "Software Cost Estimation Approaches : A Survey," pp. 824–842, 2017.

[3] B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approaches – A survey," vol. 10, pp. 177–205, 2000.

[4] V. Khatibi and D. N. . Jawawi, "Software Cost Estimation Methods : A Review," *J. Emerg. Trends Comput. Inf. Sci.*, vol. 2, no. 1, pp. 21–29, 2010.

[5] A. J. Albrecht, "Measuring application development productivity," *IBO Conference on Application Development*. pp. 83–92, 1979.

[6] B. Boehm, "Software Engineering Economics," *IEEE Trans. Softw. Eng.*, vol. 10, no. 1, pp. 4–21, 1984.

[7] C. Rush and R. Roy, "Expert Judgement in Cost Estimating: Modelling the Reasoning Process," *Concurr. Eng.*, vol. 9, no. 4, pp. 271–284, 2001.

[8] R. T. Hughes, "Expert judgement as an estimating method," *Inf. Softw. Technol.*, vol. 38, no. 2, pp. 67–75, 1996.

[9] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 11, pp. 736–743, 1997.

[10] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl, "Optimal project feature weights in analogy-based cost estimation: Improvement and limitations," *IEEE Trans. Softw. Eng.*, vol. 32, no. 2, pp. 83–92, 2006.

[11] C. Hofmeister, R. L. Nord, and D. Soni, "Global analysis: moving from software requirements specification to structural views of the software architecture," *Software, IEE Proceedings-*, vol. 152, no. 4, pp. 187–197, 2005.

[12] M. J. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Trans. Softw. Eng.*, vol. 27, no. 11, pp. 1014–1022, 2001.

[13] Y. S. Huang, C. C. Chiang, J. W. Shieh, and E. Grimson, "Prototype optimization for nearest-neighbor classification," *Pattern Recognit.*, vol. 35, no. 6, pp.

1237–1245, 2002.

[14] M. Jørgensen, “Top-down and bottom-up expert estimation of software development effort,” *Inf. Softw. Technol.*, vol. 46, no. 1, pp. 3–16, 2004.

[15] L. R. Nerkar, “Software Cost Estimation using Algorithmic Model and Non-Algorithmic Model a Review,” pp. 4–7, 2014.

[16] C. C. Model, “Basic COCOMO,” no. D, pp. 1–5, 2000.

[17] Q. Hu, “Production Functions,” vol. 23, no. 6, pp. 379–387, 1997.

[18] G. C. Low and D. R. Jeffery, “Function Points in the Estimation and Evaluation of the Software Process,” *IEEE Trans. Softw. Eng.*, vol. 16, no. 1, pp. 64–71, 1990.

[19] E. Bouwers, A. Van Deursen, and J. Visser, “Evaluating Usefulness of Software Metrics-an Industrial Experience Report,” *Proc. 35th Int. Conf. Softw. Eng.*, pp. 921–930, 2013.

[20] C. R. Symons, “Function Point Analysis: Difficulties and Improvements,” *IEEE Trans. Softw. Eng.*, vol. 14, no. 1, pp. 2–11, 1988.

[21] G. C. Low and D. R. Jeffery, “Function points in the estimation and evaluation of the software process,” *Softw. Eng. IEEE Trans.*, vol. 16, no. 1, pp. 64–71, 1990.

[22] C. F. Kemerer, “An empirical validation of software cost estimation models,” *Commun. ACM*, vol. 30, no. 5, pp. 416–429, 1987.

[23] M. V. Deshpande and S. G. Bhirud, “Analysis of Combining Software Estimation Techniques,” *Int. J. Comput. Appl.*, vol. 5, no. 3, pp. 1–2, 2010.