International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 5, October 2017, pp. 2883~2894
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i5.pp2883-2894

Insights on Research Techniques towards Cost Estimation in Software Design

Praveen Naik[1], Shantaram Nayak[2]
[1]VTU, Dept of Computer Science & Engineering, Faculty of Engg, Christ University, Bangalore, India
[2]Dept of Information Science & Engineering, RV College of Engineering Bangalore, India

Article Info

Article history:

Received Feb 16, 2017
Revised May 15, 2017
Accepted Aug 11, 2017

Keywords:

Effort estimation
Fault prediction
Software cost estimation
Software modelling

ABSTRACT

Software cost estimation is of the most challenging task in project management in order to ensuring smoother development operation and target achievement. There has been evolution of various standards tools and techniques for cost estimation practiced in the industry at present times. However, it was never investigated about the overall picturization of effectiveness of such techniques till date. This paper initiates its contribution by presenting taxonomies of conventional cost-estimation techniques and then investigates the research trends towards frequently addressed problems in it. The paper also reviews the existing techniques in well-structured manner in order to highlight the problems addressed, techniques used, advantages associated and limitation explored from literatures. Finally, we also brief the explored open research issues as an added contribution to this manuscript.

Corresponding Author:

Praveen Naik,
Research Scholar,
Dept of Computer Science & Engineering,
Faculty of Engg, Christ University, Bangalore, India.
Email: praveen.research.se@gmail.com

1. INTRODUCTION

Software development cycle plays a significant role in project management that directly impacts the productivity [1]. Normally, the process of software development involves longer duration and it suffers from certain paranoia of uncertainty [2-3]. The root cause of such uncertainty could be requirement volatility [4], defective project guidelines [5], skill gap [6], and wrong project planning [7]. An inappropriate project planning can lead to significant loss of resources with respect to time, money, and effort [8]. The prime stepping stone of success in project planning is to estimate the cost involved in the software project development. Therefore, we can define software cost estimation as the mechanism that performs forecasting the original and cumulative cost required to complete the software project on a tentative time in presence of all resources and constraints [9-10]. It is assumed to be highly complexities for many industries as cost involvement differs for different project and there are numerous factors that affect it. In this process, functional size matters the most as it is responsible for computing size of the software that is considered as the core input for majority of the modeling. One of the examples of such standard method is Common Software Measurement International Consortium (COSMIC) that is certified by ISO/IEX 19761:2011 [11]. Another essential indicator of the cost involvement is the development effort [12] that can be defined as effective amount of time used for developing the software project in progressive order. Since last decade, there has been evolution of various such mechanisms; however, none of the single mechanism can be said to 100% full proof for cost estimation [13]. Another indicator of the cost factor is line-of-code which is universally considered as the best metric to estimate cost of software development [14]. After the software is build, it is also required to be properly tested that could be done both by manual and by automated testing

[15]. There is also a cost involvement even in testing phase too and hence comprehensive testing mechanism that consumes more time and more resources are now fading slowly from the industry. According to the article presented by Smitte et al. [16], there are many reputed organization who has already initiated steps towards saving cost of software development. Irrespective of the type of work (complex software projects, test automation, remote support work, agile projects, etc), it is feasible for saving the cost provided an effective and error free modeling of cost estimation in software projects do exists.

However, there is a difference between the hype and reality. At present, there are quite a few standard techniques e.g. Constructive Cost Modeling (COCOMO) that is found exercised by various researchers for cost estimation and effort estimation. One of the biggest impediments towards modeling cost estimation techniques is to ensure accuracy in prediction rate. In order to perform prediction, the software modeling will be highly required to possess certain heuristic information [11]. In such cases, the model will be required to have access to higher number of heuristic organizational data and then its accuracy will be highly dependent on the dimension of heuristics. In other way, it can provide cost estimation but at the cost of organization resources. Hence, there are various experts who still believe that cost-estimation modeling is actually in its nascent stage and it will be quite necessary to perform re-investigation taking prominent cases of software development.

This paper discusses about the existing system, their effectiveness, limitation, in order to derive the open issues in software cost estimation. Section 1.2 discusses about the standard classification of the software cost estimation techniques in brief followed by discussion of research trends towards addressing research problem in Section 1.3. Section 2 elaborates about the existing research techniques on multiple problems highlights their advantages and limitation. Discussion of open research issues is carried out in Section 3 followed by conclusion in Section 4.

1.1. Background

Before, briefing the taxonomies of the existing software cost estimation techniques from research viewpoint, it is essential to understand certain brief detailing of it. Majority of the existing models visualizes the process of cost estimation as the function that is capable of performing computation from a evaluated cost drivers for an effective analysis [17]. The existing research work considers software requirement as the prominent cost driver. At present, there are various techniques of cost estimation techniques that can be seen from Figure 1. Basically, there are two different classified approaches i.e. algorithmic approach and non-algorithmic approach.

Algorithmic techniques are those that adopt mathematical approach where the frequently used inputs for such approaches are function point, line of codes, etc. Basically, there are three forms of algorithmic approach viz. i) Putnam's Model, ii) Constructive Cost Model, and iii) Function Point Analysis. Putnam's model performs computation of size of software using environment factor, lines of code, dispatching time of software, and build up factor of manpower. Unfortunately this model doesn't consider various potential factors of development lifecycle [18]. The second approach i.e. constructive cost model is frequently used by various researchers and is again classified into further three forms i.e. i) fundamental model, ii) intermediate model, and iii) detailed model. The prime reason for its higher adoption is its simple technique to estimate cost and its wide adoption within the industry [19]. But it also suffers from various problems e.g. non-applicability on smaller projects, estimation failures. The third algorithmic approach is Function Point Analysis that is used to measure the number of functional points available in the software projects. The prime factors in Function Point Analysis are interface file (external), logical file (internal), and varied enquiries. The interesting point of Functional Point Analysis is that it is independent from tools, language, or any specific implementation strategy. However, the pitfall of Functional Point Analysis is its manual operational technique [20]. On the other hand, non-algorithmic approaches are basically classified into 6 types (based on frequent research techniques) viz. i) top-down based, ii) Expert Judgment based, iii) Bottom-based, iv) Analogy Based, v) Artificial Neural Network (ANN), and vi) Fuzzy Logic. In top-down based approach, the global charecteristics are used for computing total cost and emphasizes more on integration. The limitation of this approach is its non-consideration of low-level problems leading to cost increment [21]. In Expert Judgment based approach, an expert's opinion is considered that is good for cases with restricted access to data. It can be used for predicting the influence on cost due to change of any underlying technologies [22]. In bottom-up based approach, the overall cost is estimated only after deriving cost of each components involved in software design. The stability of the technique is quite high but it ignores integration, documentation, etc [21]. The analogy-based approach is used for determining the parameters of existing software project as well as it also explores the equivalent old project whose parameters matches with the current project. The significant advantage of using analogy-based approach is supportability of estimator experience for assisting in enhanced cost estimation. Unfortunately, this approach is not applicable for all the software projects [23]. Both artificial neural networks as well as fuzzy logic are soft computing techniques where the former uses historical data and latter uses membership function. Although ANN offers consistency in cost estimation process buts it has maximum dependencies of training data to ensure accuracy [24]. Similarly, adoption of fuzzy logic based approach has the advantage of zero dependencies over training as well as it also offer comprehensive trustworthy cost estimation. However, even the fuzzy-logic based approach also suffers from pitfalls i.e. it offers computational complexities in many cases and moreover the accuracy completely depends on the type of the Ruleset being designed for this purpose of cost estimation [25].

Hence, it can be said that irrespective of the presence of various conventional software cost estimation techniques, there is less effective cost estimation technique for a large scale and dynamic software development process.

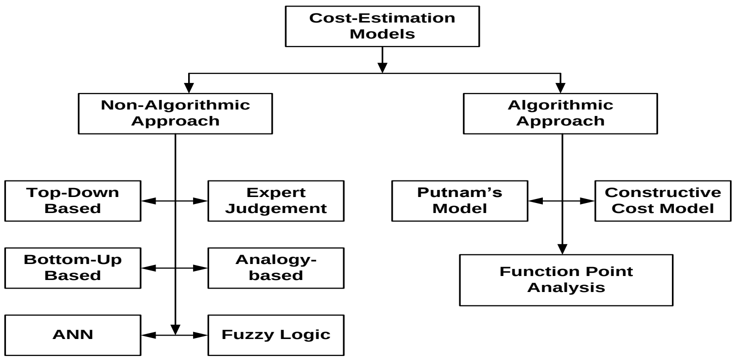

Figure 1. Taxonomies of existing software cost-estimation approaches

1.2. The Problem

This section highlights the statistics of varied number of research papers published in IEEE XPlore during duration from 2010 to till date pertaining to the problems associated with software cost estimation techniques. We reviewed the most commonly addresses problems in cost estimation techniques and graphically highlighted the current progress in it. The number of recent journals towards software cost estimation is found to be less than 50 (Figure 2), which is very less in number compared to other research problems in software modeling or engineering. The availability of relevant papers is quite less. Similar problem do existing in research towards effort estimation where only less than 25 journals are found till date (Figure 3). The trend is much reduced for fault prediction in cost estimation (Figure 4) as there are only 9 journals associated with this problem.

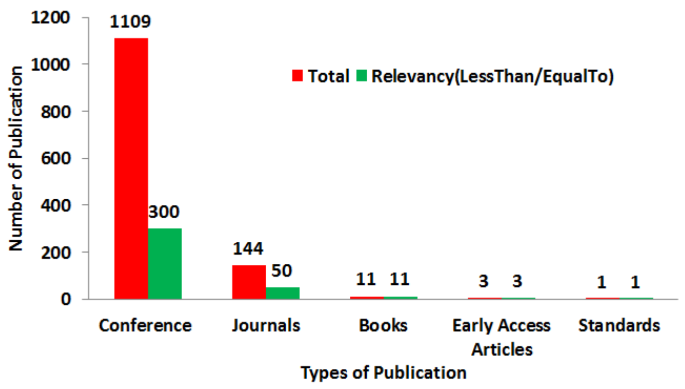

Figure 2 Research trends towards software cost estimation

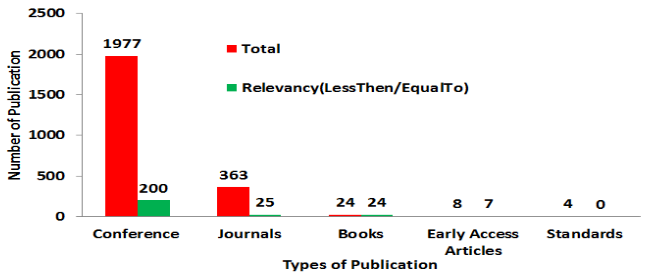

Figure 3 Research trends towards effort estimation

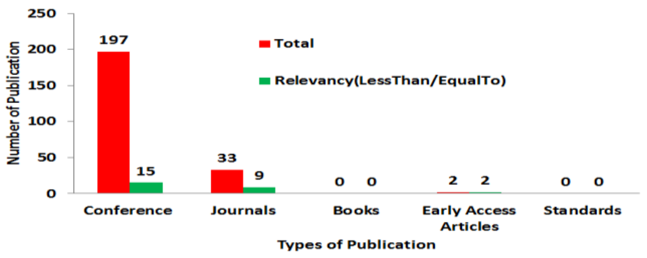

Figure 4 Research trends towards fault prediction

Apart from this, we also investigated about the trends towards other usage of other approaches of cost estimation e.g. expert judgment, analogy-based, Putnam’s model, Functional Point Analysis, COCOMO, etc. A closer look into the research trend in Figure 5 shows that Functional Point Analysis as well as COCOMO model is the most adopted techniques by various researchers till date. The proportion of the research papers using Functional Point Analysis is found to be comparatively higher than conventional COCOMO model. However, the matter of illusion is that usage of Functional Point Analysis was not adopted specifically for cost estimation problems in software and hence the relevancy rate of using Functional point analysis is quite low. Similarly, we find that although many research papers doesn’t highlights COCOMO in the preliminary section of their papers, but we find that majority (say 95%) of the researchers have adopted COCOMO model in the implementation of their approaches. Hence, the present research trend can be said to use COCOMO model with higher relevancy factor compared to usage and adoption of other conventional techniques of cost estimation.

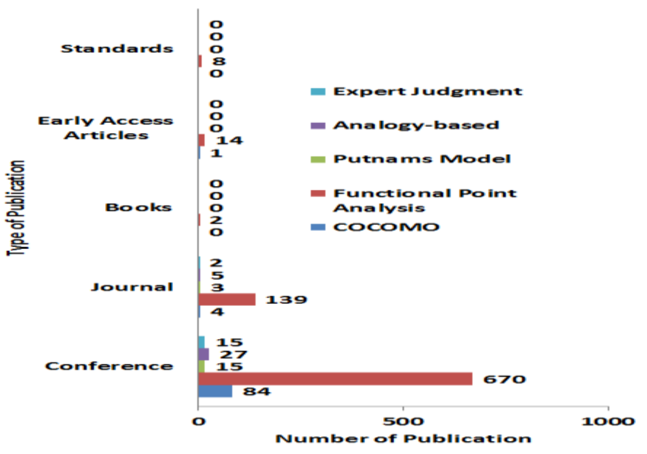

Figure 5 Research trends towards other problems

The next section elaborates about some of the frequently addressed problems and the introduced techniques from recent and relevant literatures.

2. EXISTING RESEARCH TECHNIQUES

This section discusses about the existing research techniques towards cost estimation process in the industry. As there is so much varied process of investigating the cost factor by various researchers, the discussion is carried out for the frequently used computation techniques of cost factor in the software development process. We discuss only the recent research manuscript published in the duration of 2010-till date in this regards.

2.1. General Cost Estimation Techniques

From industry viewpoint, every sector has its own kind of criterion of measuring the total number of resources involved in the process of software development. In this regards, the adoption of Functional Point Analysis can be considered as highly standard and practical one as its adoption was found in many existing corporates and firms [26]. Basically, it is one of the mechanisms to perform quantification of the active functions that are used for designing the software and it retains higher significance. But although the usage of Functional Point Analysis started back in 1979 in IBM, it suffers certain pitfalls too that are investigated by the research community. Some of the significant pitfalls are i) inconsistent outcomes, ii) mismatched points during analysis, etc. A discussion about techniques for enhancing Functional Point Analysis was presented by Junior et al. [27]. It was seen that there are various forms of enhancement being carried out towards improving the accuracy of Functional Point Analysis e.g. Fuzzy logic, Artificial Neural Network, Statistical Regression, Genetic Algorithm, interpolation technique, etc. However, according to the analysis carried out by author majority of the existing techniques do still possess certain pitfalls e.g. no clear cut solution to countermeasure existing problems, higher complexity, dependencies on current version, etc. Another standard investigation towards software cost estimation was carried out by Mittas and Angelis [28]. A statistical modeling approach has been used for investigating different forms of existing models of cost estimation. Further, a clustering mechanism was applied for unique groups. The authors have performed the experiments using datasets for measuring accuracy followed by a ranking process to understand the most elite outcome. The authors have carried out Scott-Knott test over 11 standard models to find better predictive accuracy. The cost factor of the software projects also depends on the functional size of the code. It becomes quite a significant task when it comes to embedded system design as various units of embedded system e.g. memory is significantly affected by the control unit. Study in such direction was carried out by Lind and Heldal et al. [29] by presenting a standard mechanism for estimating embedded code size called as COSMIC approach. In order to extract the precise outcomes of the study, both functional sizes as well as size of software code were subjected to correlational analysis to find that they are higher correlated. Statistical analysis was applied for this purpose. The study outcome states that there are certain prominent factors that potentially affect the estimation process e.g. consistency of the contents, clear visualization of functional domain, superior requirement quality, and precise historical data. Not only in embedded system and design, similar problems do exists in fourth generation language. The degree of complexity towards fourth generation language is quite higher and this problem is investigated by Zia et al. [30]. The technique has used empirical data on the basis of the standard specification of COCOMO II model. The study outcome has been tested using multiple tools e.g. Visual Basic, VB.Net, VC# in order to compute both actual and forecasted effort to find higher accuracy. The significant studies towards general cost estimation is quite less in terms of number of journals. Although, few significant conference manuscripts have equally contributed to share certain novel ideas towards enhancing cost estimation techniques in software design. Most recently, Hihn et al. [31] have presented a technique that can estimate cost for National Aeronautics and Space Administration (NASA) based repository data. The technique also emphasizes on potential software metric e.g. logical lines of code with varied system parameters. Using multiple mission types of space objects (earth orbiter, observatory, deep space, etc), statistical measures were applied on the basis of standard COCOMO-II specification. The technique uses analogy-based approach with various inputs of levels of inheritance, size of software, types of elements, types of mission, etc and then spectral clustering is applied for computing effort. Similar form of celestial case study was also seen in the work carried out by Cannon and Burlet [32]. The study mainly develops a prototype of flight software using C language and observation was made for time variability, composition, and density over the moon surface. The author computed cost of this software model using extrapolation approach, COCOMO using agile methodologies using source line of code. These approaches have used the standard data from the space centers and can be considered as best initiative towards cost computing models using space related data. Another unique optimization-based technique of using bio-inspired approach was seen in the work of Garg and Gupta [33]. The authors addressed the problem of accuracy pertaining to usage of agile methodologies. The problems of attribute dimensions have been minimized using Principle Component Analysis followed by identification of prominent factor with higher correlation on software design cost. The study outcome is completely independent of historical data or any form of other information. The study has evaluated coefficient value of various test components e.g. size of team, complexity involved in projects, type of application, total functional points, platform used in development, etc. The practice of applying optimization technique was also seen in the work of Gharehchopogh et al. [34]. The author has applies Tabu search algorithm as well as genetic algorithm for this purpose. The technique uses NASA-related dataset along with COCOMO model to find reduced error in outcomes. Similar author has also applied Artificial Bee Colony along with genetic algorithm [35]. Adoption of Fuzzy Logic was seen in the work of Idri and Zahi [36] considering the dataset of web software. Liu et al. [37] have also used Particle Swarm Optimization in order to increase the accuracy of cost estimation for analogy based approach. Uniquely, the author considers a factor called as *Non-Orthogonal Space Distance* for estimating the similarity measures of the project. Sarno et al. [38] have used neural network for similar cause. Enhancement of analogy-based approach was seen in the study of Phannachitta et al. [39] using probability-based computation of distance function for cost computation.

Table 1. Summary of the Existing Techniques of Cost Estimation

Authors	Problems	Techniques	Advantages	Limitation
Junior [27]	Accuracy problem of Functional Point Analysis	Qualitative Study	Highlighted techniques to improve	-no clear cut solution -higher complexity -dependencies on current version
Mittas [28]	Predictive Accuracy	Statistical Measures (ANOVA)	-Extensive Assessment -	-doesn't consider dynamic or uncertainty factor in modeling
Lind [29]	Estimating code size in embedded system	-COSMIC -Statistical Measures	-Simple & Practical Framework design	-Doesn't discuss the applicability -No benchmarking
Zia [30]	Fourth Generation language	-cross-validation -Empirical Approach	-Simple modeling approach	-doesn't consider dynamic or uncertainty factor in modeling
Hihn [31]	NASA Data	-Statistical (f-test, t-test) - spectral clustering	-Capable of considering complex data	-No benchmarking
Cannon [32]	Cost estimation for lunar data	COCOMO, Extrapolation, Source Lines of Code, Agile	-Effective Analysis technique -Offer good estimates	-No benchmarking or validation
Garg [33]	Accuracy of Agile	Principle Component Analysis	-No dependency over historical data	-No benchmarking or validation
Gharehchopogh [34]	Cost estimation	Genetic Algorithm, Tabu Search, COCOMO	-Reduced error in outcomes	-Doesn't discuss computational complexity
Soleimanian [35]	Cost estimation	Genetic Algorithm, Artificial Bee Colony, COCOMO	-Reduced error in outcomes	-Doesn't discuss computational complexity
Idri [36]	Comparing analogy-based approach	Fuzzy Logic	-Applicable for web applications	-No benchmarking -Doesn't discuss computational complexity
Liu [37]	Accuracy in cost estimation	Particle Swarm Optimization	-Fits to real-world projects -Reduced Error	-Doesn't discuss computational complexity
Sarno [38]	Accuracy in cost estimation	-Feed-forward neural network -back propagation -COCOMO	-Reduced error -Simple implementation	-No benchmarking -Convergence performance not discussed
Phannachitta [39]	Error reduction in analogy-based approach	Probabilistic Approach	-Faster implementation -tested over multiple dataset	-No benchmarking -Convergence performance not discussed

2.2. Effort Estimation Technique

Effort estimation is the direct functional task to be carried out towards majority of the cost-estimation model till date. Although, there are various tools and technique to compute effort factor within an industry, but still there are certain open challenges towards implementing it e.g. inferior knowledge management, non-adaptability on standard estimation techniques, tool-related problems, negligence of project performance parameters [40]. Various challenges pertaining to the techniques of effort estimation at present are found to be human resources, governance, tools, and processes. At present, there is no such model or framework that overall addresses such problems jointly. Most recently, there is investigation towards effort estimation in software design in early stages for better consideration of probable cost involved within this. One such technique has been presented by Satapathy et al. [41], where the authors have applied frequently used machine learning approach known as *random forest*. The technique uses random forest for optimizing the accuracy in the predictive outcome. The study also considers various forms of the complexity factors e.g. portability, changeability, concurrency, efficiency of end user, code reusability, security features etc. The authors have also performs analysis over the error, proximities, complexities, and outliers. The study outcome was assessed using standard performance parameters e.g. mean squared errors, prediction accuracy, mean magnitude of relative error. Study towards development efforts and its estimation techniques have been also presented by Bardsiri et al. [42] towards the aim of enhancing the accuracy level. The authors have applied jointly multiple techniques e.g. analogy-based techniques, fuzzy logic, and neural network. The study outcome shows impressive percentage of prediction accuracy. It was observed that much of the emphasis was laid on knowledge discovery process using mining techniques in effort estimation techniques. Applying mining approach can incorporate further predictive feature in software modeling. Dejaeger et al. [43] have carried out a discussion where the importance of mining approaches have been made. A statistical approach is applied for performing comparative analysis on various techniques. Grbac et al. [44] have carried out a study towards effort estimation during inspection. The special consideration is about the early stage of the development process. The outcome of the study suggests that an effective decision factor significantly assists in enhancing the product reliability. Another bigger challenge in the area of software engineering the dispersion of the falsified information that leads to declination of the product quality as well as error-prone effort estimates. Study towards such problem was discussed by Jorgensen and Grimstad [45] where the authors have performed an experiment considering various organizational data. The prime purpose of the presented study is to perform cost computation of the effort being used for estimating unnecessary task. This technique plays much important role for any organization to find original effort towards productivity and not towards unnecessary trials of work. The authors have used quantitative-based methodology using statistics in order to carry out the objectives. A significant problem with effort estimation is less involvement of more number of effective quality parameters. Study towards this problem was addressed by Azath and Wahidabanu [46] using standard COCOMO model along with usage of ISO 9126 standards and its compliant quality factors. The authors have also used functional points and its outcome is assessed using total degree of influence, usability, reliability, maintainability, and efficiency. Similarly, the factors involved in effort estimation will need a precise combination of different approach to make a suitable balance between simplicity and sophistication. Therefore, study emphasizing on contents and parameters required for effort estimation was carried out by Kocaguneli et al. [47]. The author uses spatial-based approach between the instances and features that further assists in effort estimation. A CART-based approach as well as COCOMO is also utilized for active learning in order to perform predictive modeling. The same authors [48] have also performed investigation towards 10 different datasets using analogy-based approach for effort estimation. Kocaguneli et al. [49] have also discussed that it is not necessary to select the best estimation method out of many method but to use the methods intermittently. The author concludes that selection of one method should be discouraged and inspite a combination of different method can best suit the purpose of effort estimation.

Table 2 Summary of the existing Techniques of Effort Estimation

Authors	Problems	Techniques	Advantages	Limitation
Satapathy et al. [41]	Prediction Accuracy	-Random Forest -Use Case Points	-Higher prediction accuracy	-computational complexity not analyzed.
Bardsiri et al. [42]	Increasing accuracy	-Analogy-based Techniques -Fuzzy Logic -Artificial neural network	-Higher prediction accuracy -Applicable on large dataset	-Lack of uncertainty factor -Stability factor not found
Dejaeger et al. [43]	-Comparative Study	-different mining approach	-regression and COCOMO are best suited	-Highly dependent on expert's judgment
Grbac et al. [44]	-value quantification during inspection	-expert judgment -Statistical Measures	-increased in quality of product	-No effective benchmarking
Jorgensen and Grimstad [45]	-falsified effort estimation	-Statistical Measures	-Simplified modeling	-overall outcome cannot be generalized to other scenario
Azath and Wahidabanu [46]	-effort estimation	-Functional point -COCOMO -ISO 9126	-very simple implementation approach	-Doesn't address the uncertainty/dynamic problems
Kocaguneli et al. [47]	-effort estimation	-COCOMO -CART -Euclidean distance	-offer better suggestion for choice of estimation techniques	-No benchmarking -Doesn't discuss about computational complexity.
Kocaguneli et al. [48]	-Effort estimation	-Analogy-based estimation -selection of nearest neighbor	-Improved system performance -Reduced error rate	-No benchmarking -Doesn't discuss computational problems
Kocaguneli et al. [49]	-Effort estimation	Combination of multiple methods	-Reduced error rate	-No benchmarking -Doesn't discuss computational problems

2.3. Fault Estimation Techniques

When the software is designed with certain error prone planning, it is ought to possess certain level of fault or defect. For this reason, the testing and quality assurance members are responsible for identifying a bug and remove it. However, the concept of cost computation evolves in estimating faults residing in software design. Although, majority of the existing mechanism of fault detection only works after the fault is really incorporated within the software, by that time, it becomes too late to debug it. Sometime is even a very high expensive matter to remove the faults. Hence, the concept of predictive principle in cost estimation is equally helpful for fault detection which is definitely in the early stage. The advantage is to identify the possible faults even before it could possibly inflict some serious problems in the software design. Study addressing such problem can be found in the work carried out by Fagundes et al. [50] where a predictive principle is introduced by the author. The technique jointly uses both parametric as well as non-parametric frameworks for enhancing the accuracy of fault prediction. Considering the NASA dataset, the authors have carried out experiments using machine learning language. The study outcome was found to have better capability of identifying the non-faulty to faulty methods. Ucan et al. [51] introduced an estimation processes to find the cost of “detecting defect” in the instrumented programs which may work traditionally or could be supported with a collaborative platform. It is found that the collaborative associate platform given defect of 3.76 per hour as compare to the traditional process, where it is 4.07. This work may influence the thought of web based enterprise application to consider the parameters that system having collaborative approach of development or not by introducing some factor say α (where, the value of α will be higher if no collaborative tool used for development). The parameter α reduced the cost of development. The adoption of collaborative platform is coming mandatory as the software production process uses various models of out-sourcing, in-sourcing, off-shore development etc, where people are located at different geographical region and not possible to have front to front ad-hoc discussions. Sometimes hardware as well as software-based design and their associated components could be responsible for software failures. Study in such direction was carried out by Ozcelik and Yilmaz [52] where a predictive system is design that can estimate the amount of faults in the software. The evaluation of the study was carried out using a standard transformation tool in presence of multiple defects while study outcomes are evaluated using performance parameters of prediction accuracy. Predictive study towards fault computation was also carried out by Liu et al. [53] where the emphasis is mainly laid over the classification process. The technique makes use of both feature as well as reduction of number of instances as a part of data preprocessing. An analysis of relevance is carried out during the selection process of feature. It than performs controlling of redundancies using threshold-based clustering approach. Similar line of work was also carried out by Dejaeger et al. [54] by introducing a Bayesian network classifier. The authors have also investigated the feasibility of implementing Markov blanket principle with respect to Bayesian network theory. Similar NASA dataset has been used for determining line of code metrics, Halstead metrics, and McCabe metric. Monden et al. [55] have presented a technique for predicting fault as well as assessing the cost estimates out of it. With an aid of software testing, the proposed system evaluates the possible allocation techniques for effort estimation depending on the outcome of the faults. Arcuri et al. [56] have presented a technique that uses random testing in order to detect the fault in the software design. The reason behind adaption of random testing is its supportability of large number of features. Bishnu and Bhattacherjee [57] have introduced a classification technique for performing fault prediction. The authors have used K-means clustering algorithm in order to evaluate the cluster quality and predicting the possible faults in the code. Jin et al. [58] have presented a technique using support vector machine as well as neural network for assisting in selection of parameters that would positive affect the fault prediction. Database from NASA is used with multiple metrics in order to perform cross-validation followed by accuracy testing. The study outcome was found to possess a potential link between fault factors and software metrics. Software testing has also significantly gained higher degree of attention among the researchers. In accurate testing strategies will essentially lead to generation of faults and thereby it negatively affects the economy of an organization. Study towards strengthening the software testing was also carried out Ramos et al. [59]. The authors have introduced a model that significantly assists in ensuring lesser amount of defects as well as faults in the software design. It also assists in overcoming the skill gap among the testers. This competence model can be used by any organization.

Table 3 Summary of Existing Techniques of Fault Estimation

Authors	Problems	Techniques	Advantages	Limitation
Fagundes et al. [51]	Fault prediction	-Joint usage of parametric & non-parametric techniques	-fitting model to set of data points	-Computational complexity not assessed.
Ucan et al. [59]	Efficiency in Defect detection	-collaborative virtual environment	-faster response	-No benchmarking -Space complexity is not addressed.
Ozcelik and Yilmaz [53]	Runtime prediction accuracy	CIL tool, Weka, PAPI tool,	Reduced runtime overhead	-No benchmarking
Liu et al. [54]	Classification problems	Data preprocessing, random under-sampling	-maintain better balance between instances	-No benchmarking -Doesn't address the dynamic problems
Dejaeger et al. [55]	Fault prediction	Bayesian network classifier	-balanced predictive and comprehensibility performance	-Complex relationship in both dynamic and static features of code not considered.
Monden et al. [56]	-Forecasting fault -Allocation of effort	-Empirical approach -Simulation	-cost effective prediction of fault	-higher dependencies on internal resources.
Arcuri et al. [57]	Fault detection	Random testing	-supportability of higher features	-N/A-
Bishnu and Bhattacherjee [58]	Fault prediction	k-means clustering	-better performance of error rates	-no applicable on dynamic programs
Jin et al. [59]	Fault prediction	-Neural network -Support vector Machine	-No extra cost towards implementation -no dependency towards experts knowledge	-training complexities associated is not discussed
Ramos et al. [D10]	Minimizing faults	-Competence model	-practically affect users to minimize faults	-Outcome applicability is narrow for large scale applications

3 OPEN RESEARCH ISSUES

From the prior section, it was seen that there are various techniques towards ensuring that cost estimation in the software design. It has been explored that the existing techniques of software cost estimation have highly diversified strategies to implement but unfortunately, it is found with higher number of challenges. All the techniques have their own advantages as well as limitations too. However, there are certain set of problems that has been found not to be addressed by the existing research work.

- **Less focus on Input:** Basically 99% of all the existing research papers have adopted dataset e.g. NASA, where the existing frameworks considers input to be i) functional point, ii) line of source code, iii) instruction set, etc. Such technique doesn't offer an accurate outcome when it is applied for estimating cost for the software development. Such scenarios are highly static and uniform and thereby its outcome cannot be said to be applicable for dynamic environment. Hence, there was a lesser emphasis towards ensuring more precise inputs while estimating cost.
- **No Studies towards Computational Efficiency:** Existing techniques have mainly introduced cost computation method on different dataset. There are also studies where large numbers of datasets are used with more inclusion of recursive function. Such types of approaches significantly increases both time and space complexity. Some of the studies have been claimed to offer reduced processing burden, but exact validation of it is missing in the literature. Without ensuring cost effective computation, the applicability of such cost computation techniques over low-powered devices are questionable.
- **More Generalized and Less Specific:** The existing techniques have not taken case studies of applicability of cost estimation model on particular software. All the current investigations use the term *software* as a very generic model with no discrete elaboration of the system. Because of this reason, it is not sure whether the existing cost-computing models are really applicable in real-world application. At present, there are more than lakhs of software with thousand numbers of technologies running over it. It is well known that in present time's usage of wireless-based products are in increasing trends. Unfortunately, there is no such study on cost computation model that deals with wireless-based products.
- **No Network Consideration:** Majority of the existing system uses the term software and doesn't consider anything else. In reality, it is known that maximum software is now connected with the advent of pervasive and ubiquitous computing. Hence, ignoring the network statistics and proceeding towards cost computational model for software will be impartial work. It is necessary that all the connected components of the software e.g. user, network, etc. should be considered while working over the cost computation model.
- **Few Studies towards Minimization:** Cost is a variable in the area of software project management. An effective project planning always calls for reduction of the cost involved in it. Unfortunately, the existing

techniques are more focus on cost estimation using different dataset and less towards cost minimization. Study towards an effective optimization technique will always results in minimal cost involvement. This theoretical viewpoint was not seen in many studies towards software modeling and cost estimation techniques in present times.

- **Few Standard Modeling towards Cost Estimation:** A closer look into the outcomes of any of the recent studies towards cost estimation will show very few benchmarked techniques. Usage of COCOMO model is definitely not without flaws e.g. more prone to adopt backdated waterfall model, more experimental and less calibrated, non-supportability of duration calculation in small scale software projects. Hence, even after knowing such pitfalls, majority of the work has been using COCOMO irrespective of taking a chance to introduce a new mathematical modeling or algorithmic approach.

3. CONCLUSION

This paper has discussed the research traits of the software cost estimation. There are various conventional models for performing cost estimation explored in recent time e.g. algorithmic and non-algorithmic approach. Although both the approaches has associated advantages as well as limitation, but still they are in practice at large. We also discuss the existing research trends to find that there are very less number of good research work being carried out towards software cost estimation, effort estimation, fault prediction, etc. We find that COCOMO is the backbone of majority of the existing research techniques. Our findings are i) there are very less benchmarked research methodology that can be recommended in present day, ii) not a single cost-estimation technique can be referred as a robust one, however, a combination of multiple techniques (hybridizing) can show better result, iii) none of the existing techniques are really found to address computational complexity problems, iv) there are various open research issues pertaining to existing literatures of cost-estimation approach, which will require serious consideration, etc. Hence, our future direction of the research work will be towards bridging the gap of existing system.

We have noticed that almost all the existing research work has use the term software in generic sense whereas in really there are diverse forms of it with manifold complexities. Hence, we will like to adopt a specific case study of software and make it more discrete. It is because it is not sure if any of the existing technique can be applicable in any specific software application. Hence, our first future work will be to consider a software that assists in wireless communication and the compute a cost pertaining to development of it. Our second level of future work will be to re-think on the term software and make it more applicable to the present time application and present a cost-estimation model for it. We will like to investigate about the usage of high frequency signal and their utilization over the software for civic benefits in order to see the cost involvement in it. Our final level of future work will be to ensure that our cost involvement could be possibly minimized while retaining higher product efficiency. In short, our future direction of the work will be to give a practical shape to software using a case study of it so that it becomes easy for the research community to understand the applicability of my framework.

REFERENCES

[1] Sudhakar, "Software Development Teams: Performance," *productivity and innovation, phi learning pvt. ltd*, 2015
[2] H. Mohanty, J. R. Mohanty, A.K. Balakrishnan, "Trends in Software Testing," *Springer-Computers*, 2016
[3] J. McManus, "Risk Management in Software Development Projects," *Routledge*, 2012
[4] A. P. Sage, W. B. Rouse, "Handbook of Systems Engineering and Management," *John Wiley & Sons*, 2009
[5] S. Jha, "The Project Manager's Communication Toolkit", *CRC Press*, 2010
[6] B. Tulgan, "Bridging the Soft Skills Gap: How to Teach the Missing Basics to Todays Young Talent," *John Wiley & Sons*, 2015
[7] P. Serrador, "Project Planning and Project Success: The 25% Solution," *CRC Press*, 2014
[8] T. Melton, "Real Project Planning: Developing a Project Delivery Strategy," *Butterworth-Heinemann*, 2011
[9] C. Artigues, S. Demassey, E. Neron, "Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications," *John Wiley & Sons*, 2013
[10] R. Klein, "Scheduling of Resource-Constrained Projects," *Springer Science & Business Media*, 2012
[11] C. Ebert and H. Soubra, "Functional Size Estimation Technologies for Software Maintenance," *in IEEE Software*, vol. 31, no. 6, pp. 24-29, Nov.-Dec. 2014.
[12] M. Jørgensen, "Relative Estimation of Software Development Effort: It Matters with What and How You Compare," *in IEEE Software*, vol. 30, no. 2, pp. 74-79, March-April 2013.
[13] M. Jørgensen, "What We Do and Don't Know about Software Development Effort Estimation," *in IEEE Software*, vol. 31, no. 2, pp. 37-40, Mar.-Apr. 2014.
[14] E. Morozoff, "Using a Line of Code Metric to Understand Software Rework," *in IEEE Software*, vol. 27, no. 1, pp. 72-77, Jan.-Feb. 2010.

[15] M. Polo, P. Reales, M. Piattini and C. Ebert, "Test Automation," *in IEEE Software,* vol. 30, no. 1, pp. 84-89, Jan.-Feb. 2013.
[16] D. Smite, F. Calefato and C. Wohlin, "Cost Savings in Global Software Engineering: Where's the Evidence?," *in IEEE Software*, vol. 32, no. 4, pp. 26-32, July-Aug. 2015.
[17] R.V. Ray, J.K. Pinto, "Cost and Value Management in Projects," *John Wiley & Sons*,2011
[18] R. Mall, "Fundamentals of Software Engineering," *PHI Learning Pvt. Ltd*, 2014
[19] B. B. Agarwal, S. P. Tayal, M. Gupta, "Software Engineering and Testing," *Jones & Bartlett Learning*, 2010
[20] J. Engelhart, P. Langbroek, "Function Point Analysis (FPA) for Software Enhancement," *Nesma*, 2009
[21] S.L.Pfleeger, F. Wu, R. Lewis, "Software Cost Estimation and Sizing Methods: Issues, and Guidelines," *Rand Corporation*, 2005
[22] A. Oram, G. Wilson, "Making Software: What Really Works, and Why We Believe It," *O'Reilly Media, Inc*, 2010
[23] G.K. Mislick, D. A. Nussbaum, "Cost Estimation: Methods and Tools", *John Wiley & Sons*, 2015
[24] F. S. Gharehchopogh, "Neural Networks Application in Software Cost Estimation: A Case Study," *International Symposium on Innovations in Intelligent Systems and Applications, Istanbul*, pp. 69-73, 2011
[25] N. Kushwaha and Suryakant, "Software Cost Estimation Using the Improved Fuzzy Logic Framework," *Conference on IT in Business, Industry and Government (CSIBIG), Indore*, pp. 1-5, 2014
[26] "Internal Year of Software Measurement", http://www.ifpug.org/about-ifpug/about-function-point-analysis/, Retrived on 12th May-2017
[27] M. de Freitas Junior, M. Fantinato and V. Sun, "Improvements to the Function Point Analysis Method: A Systematic Literature Review," in *IEEE Transactions on Engineering Management*, vol. 62, no. 4, pp. 495-506, 2015.
[28] N. Mittas and L. Angelis, "Ranking and Clustering Software Cost Estimation Models through a Multiple Comparisons Algorithm," in *IEEE Transactions on Software Engineering*, vol. 39, no. 4, pp. 537-551, 2013.
[29] K. Lind and R. Heldal, "A Practical Approach to Size Estimation of Embedded Software Components," in *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 993-1007, 2012.
[30] Z. Zia, A. Rashid and K. U. Zaman, "Software Cost Estimation for Component Based Fourth-generation-language Software Applications," in *IET Software*, vol. 5, no. 1, pp. 103-110, 2011.
[31] J. Hihn, L. Juster, J. Johnson, T. Menzies and G. Michael, "Improving and expanding NASA Software Cost Estimation Methods," *IEEE Aerospace Conference*, Big Sky, MT, pp. 1-12, 2016
[32] H. Cannon and K. Gundy-Burlet, "Software Cost Estimation for the LADEE Mission," *IEEE Aerospace Conference*, Big Sky, MT, pp. 1-8, 2015
[33] S. Garg and D. Gupta, "PCA Based Cost Estimation Model for Agile Software Development Projects," *International Conference on Industrial Engineering and Operations Management (IEOM)*, Dubai, pp. 1-7, 2015
[34] F. S. Gharehchopogh, R. Rezaii and B. Arasteh, "A New Approach By Using Tabu Search and Genetic Algorithms in Software Cost Estimation," *2015 9th International Conference on Application of Information and Communication Technologies (AICT)*, Rostov on Don, pp. 113-17, 2015
[35] F. S. Gharehchopogh, I. Maleki and A. Talebi, "Using Hybrid Model of Artificial Bee Colony and Genetic Algorithms in Software Cost Estimation," *9th International Conference on Application of Information and Communication Technologies (AICT)*, Rostov on Don, pp. 102-106, 2015
[36] A. Idri and A. Zahi, "Software Cost Estimation by Classical and Fuzzy Analogy for Web Hypermedia Applications: A Replicated Study," *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Singapore, pp. 207-213, 2013
[37] Q. Liu, X. Chu, J. Xiao and H. Zhu, "Optimizing Non-orthogonal Space Distance Using PSO in Software Cost Estimation," *IEEE 38th Annual Computer Software and Applications Conference*, Vasteras, pp. 21-26, 2014
[38] R. Sarno, J. Sidabutar and Sarwosri, "Comparison of Different Neural Network Architectures for Software Cost Estimation," *International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, Bandung, pp. 68-73, 2015
[39] P. Phannachitta, J. Keung, A. Monden and K. i. Matsumoto, "Improving Analogy-Based Software Cost Estimation through Probabilistic-Based Similarity Measures," *20th Asia-Pacific Software Engineering Conference (APSEC)*, Bangkok, pp. 541-546, 2013
[40] D. Basten and T. Hoerstrup, "Organizational Effort Estimation," in *Computer*, vol. 47, no. 8, pp. 76-79, Aug. 2014
[41] S. M. Satapathy, B. P. Acharya and S. K. Rath, "Early Stage Software Effort Estimation Using Random Forest Technique Based on Use Case Points," in *IET Software*, vol. 10, no. 1, pp. 10-17, 2 2016
[42] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim and E. Khatibi, "Increasing the Accuracy of Software Development Effort Estimation Using Projects Clustering," in *IET Software*, vol. 6, no. 6, pp. 461-473, 2012
[43] K. Dejaeger, W. Verbeke, D. Martens and B. Baesens, "Data Mining Techniques for Software Effort Estimation: A Comparative Study," in *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 375-397, 2012
[44] T. G. Grbac, Z. Car and D. Huljenic, "Quantifying Value of Adding Inspection Effort Early in the Development Process: A Case Study," in *IET Software*, vol. 6, no. 3, pp. 249-259, 2012
[45] M. Jorgensen and S. Grimstad, "The Impact of Irrelevant and Misleading Information on Software Development Effort Estimates: A Randomized Controlled Field Experiment," in *IEEE Transactions on Software Engineering*, vol. 37, no. 5, pp. 695-707, S, 2011.
[46] H. Azath and R. S. D. Wahidabanu, "Efficient Effort Estimation System Viz. Function Points and Quality Assurance Coverage," in *IET Software*, vol. 6, no. 4, pp. 335-341, 2012
[47] E. Kocaguneli, T. Menzies, J. Keung, D. Cok and R. Madachy, "Active Learning and Effort Estimation: Finding the Essential Content of Software Effort Estimation Data," in *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1040-1053, 2013

[48] E. Kocaguneli, T. Menzies, A. Bener and J. W. Keung, "Exploiting the Essential Assumptions of Analogy-Based Effort Estimation," in *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 425-438, 2012

[49] E. Kocaguneli, T. Menzies and J. W. Keung, "On the Value of Ensemble Effort Estimation," in *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1403-1416, 2012.

[50] R. A. A. Fagundes, R. M. C. R. Souza and F. J. A. Cysneiros, "Zero-inflated Prediction Model in Software-fault Data," in *IET Software*, vol. 10, no. 1, pp. 1-9, 2016.

[51] J. P. Ucan, O. S. Gomez and R. A. Aguilar, "Assessment of Software Defect Detection Efficiency and Cost Through an Intelligent Collaborative Virtual Environment," in *IEEE Latin America Transactions*, vol. 14, no. 7, pp. 3364-3369, 2016.

[52] B. Ozcelik and C. Yilmaz, "Seer: A Lightweight Online Failure Prediction Approach," in *IEEE Transactions on Software Engineering*, vol. 42, no. 1, pp. 26-46, 2016.

[53] W. Liu, S. Liu, Q. Gu, J. Chen, X. Chen and D. Chen, "Empirical Studies of a Two-Stage Data Preprocessing Approach for Software Fault Prediction," in *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 38-53, 2016.

[54] K. Dejaeger, T. Verbraken and B. Baesens, "Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers," in *IEEE Transactions on Software Engineering*, vol. 39, no. 2, pp. 237-257, 2013

[55] A. Monden *et al*., "Assessing the Cost Effectiveness of Fault Prediction in Acceptance Testing," in *IEEE Transactions on Software Engineering*, vol. 39, no. 10, pp. 1345-1357, 2013.

[56] A. Arcuri and L. Briand, "Formal Analysis of the Probability of Interaction Fault Detection Using Random Testing," in *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1088-1099, 2012.

[57] P. S. Bishnu and V. Bhattacherjee," Software Fault Prediction Using Quad Tree-Based K-Means Clustering Algorithm," *IEEE transactions on knowledge and data engineering*, vol. 24, no. 6, 2012

[58] C. Jin S.-W., Jin J.-M. Ye1, "Artificial Neural Network-based Metric Selection for Software Fault-prone Prediction Model,*" IET Software,* 2012

[59] J. Saldan-Ramos A. Sanz-Esteban J. Garcıa-Guzman A. Amescua, "Design of a Competence Model for Testing Teams," *IET Software*, 2011