Journal of Engineering and Applied Sciences 14 (11): 3643-3650, 2019
ISSN: 1816-949X

A Model Combining a Priori and Posteriori Contexts to Estimate the Costs of a Software Project

[1]Mua'ad Abu-Faraj, [2]Khalid T. Al-Sarayreh and [3]Kenza Meridji
[1]Department of Computer Information Systems, Faculty of Information Technology and Systems, University of Jordan, 77110 Aqaba, Jordan
[2]Department of Software Engineering, Faculty of Prince, Al-Hussein Bin Abdallah II for Information Technology, Hashemite University, 13133 Zarqa, Jordan
[3]Department of Software Engineering Faculty of Information Technology, University of Petra, 11196 Amman, Jordan

Abstract: An early cost estimation model is one of the important issues in software engineering projects. It has been observed that cost estimation software projects often differ from the final costs. This study presents a proposed model combining between a priori and posteriori contexts for early estimation of the cost in order to avoid the used confusion of these two estimations concepts. This research contributes to research knowledge by proposing and validating most of the common cost estimation models used throughout the a priori and posteriori contexts and it highlights some of the confusion of such estimation models uses in software engineering development projects. In addition, this study lists some of the estimation techniques that can be used in a priori and posteriori contexts. The proposed model in this study can be used to help organizations to deliver their projects on time and with estimated budget in a context of fixed-price agreements. Therefore, organizations must conclude well in advance such estimates which are called the a priori cost estimation. Whilst these organizations typically use estimation techniques based either on informal opinions or organizational experience or on a posteriori estimation models developed internally or developed by outside consultancy.

Key words: A priori context, a posteriori context, cost estimation models, informal opinions, organization, software

INTRODUCTION

Software project estimation models represent a challenge to most organizations and to their customers who suffers from the cost development of software projects significantly over budget with significant delays in schedules as well as with less functionality than guaranteed and with unknown levels of quality. Estimation and quality are the most common issues facing managers and practitioners as well as their users and organizations (Al-Sarayreh and Meridji, 2013).

Cost estimation methods are the process of predicting the effort required to develop systems, accurate cost estimations are crucial to both developers and customers. The size measure of software development depends on human effort and the most cost estimation methods focuses on this aspect and it gives the estimation in terms of person-months or hours. Most cost estimation models are based on the size measure and the accuracy of size estimation which directly affects the accuracy of cost estimation. Most cost estimation models attempt to produce an effort estimates which can then be converted into cost and duration.

The computation of cost estimation is done by based on a rough calculation of the obtainable information such as the total cost of a product, project or agenda. Cost estimation models are categorized into algorithmic and non-algorithmic, each with strengths and weaknesses, although, the accuracy of the estimates are the key aspect in choosing the cost estimation model.

Fundamentally, there are two kinds of cost estimation contexts in software engineering: a priori and a posteriori contexts. A priori and a posteriori are terms used in view point to make a distinction between two different types of knowledge such as: rationalization and argument: a priori knowledge is known without having any practice and posteriori knowledge need to be proved all the way through practice (Al-Sarayreh and Meridji, 2013).

Many estimation models are built and based on data from precedent projects, this corresponds to a posteriori context, nevertheless, these models are used usually in an a priori context and in early development life cycle. Generally, in software engineering, the a posteriori models would not have been verified in an a priori context. There

Corresponding Author: Mua'ad Abu-Faraj, Department of Computer Information System, Faculty of Information Technology and Systems, University of Jordan, 77110 Aqaba, Jordan

is mystification in some published studies as well as some used estimation tools like ONTOCOM, COTS and COCOMO about the concepts of a priori and a posteriori contexts.

Some of the reasons behind these problems are (Al-Sarayreh and Meridji, 2013): many estimation models are built and based on data from precedent projects: this corresponds to an a posteriori context, though, these models are classically used in an a priori context and quite early in the development life cycle. Usually, the a posteriori models would not have been demonstrated in an a priori context.

Most cost estimation approaches have little algebraic basis and have not been validated. The consistency of inputs to cost estimation models varies extensively and this could reduce the validity of historical data as a basis for justification of these models.

There are no studies differentiating between the use of a priori and a posteriori estimation context in software engineering for the development process of renewable energy projects, nor models of combination between a priori and posteriori contexts for early estimating cost to avoid the confusion use of these two estimation concepts.

LITERATURE REVIEW

In the software engineering literature, there is a lot of work claiming to address early estimations but a closer study of these studies indicates that they are applying a posteriori context as an a priori context. On one hand, there are many early estimation methods that could be used in a priori contexts such as the ones listed by Kitchenham *et al.* (2002) for example: average, CA-estimacs, comparison, proportion and widget counting and delphi methods. While Nelson (1967) introduced early estimating model could be used in a priori context.

On the other hand, there are many other estimation techniques and models constructed and developed from information available after the completion of projects (Grimstad and Jorgensen, 2006) such models should be qualified as a posteriori estimation models such as COCOMO (Putnam and Myers, 1991), SLIM (Jensen in 1997), Checkpoint (Putnam, 1978), PRICE-S (Park in 1988), SEER (Cuadrado-Gallego *et al.*, 2006), Walston-Felix Model (Song *et al.*, 2007), Bailey-Basili Model (Stellman and Greene, 2005), Boeing Model (Summerville, 2004), Doty Model for KLOC (Bochm and Abts, 2000), Albrecht and Gaffney Model (Mendes *et al.*, 2005), Kemmerer Model (Mohagheghi *et al.*, 2005), Matson, Barnett and Mellichamp Model (Wieczorek, 2002).

Cost modeling in software engineering was initiated with the Software Development Corporation (SDC) study of 104 attributes of 169 software projects (Boehm and Abts, 2000). This led to some useful partial models in the late 1960s and early 1970s.

The late 1970s was a high point of new models such as SLIM, Checkpoint, PRICE-S, SEER and COCOMO. The majority of these researchers started working at the same time on developing models of cost estimation. They all faced the same problems like: software grew in size and in complexity, making it very difficult to estimate accurately the cost of software development.

There are only a few studies for finding the estimated cost of projects in the early stages that followed a priori context in software engineering, for example (Kitchenham *et al.*, 2002). The majority of the estimation models built based on effort after the completion of the projects or the comparison with other similar projects, for example, ONTOCOM (Simperl *et al.*, 2006) where the researchers introduced a priori cost model but move on to a posteriori approach to complete their estimation.

TYPES OF ESTIMATION MODELS

Estimation models are essential for effective software engineering for RE project and their management. During the past four decades, many cost estimation techniques have been proposed to predict the cost. A common weakness of most models is their limited helpfulness to predict the cost exactly at an early stage of the development life cycle. To be aware of this problem, there are primarily two types of estimation models: a posteriori estimation models and a priori estimation models (Al-Sarayreh and Meridji, 2013).

A posteriori estimation models: Estimation models are still immature field of knowledge in software engineering wherever most of the estimation techniques and models proposed to practitioners have never been verified autonomously on past projects and for many of the models that have been built with precedent projects, most still have a low degree of precision in their estimation (Al-Sarayreh and Meridji, 2013).

Software engineers and managers who are using a posteriori estimation models should also be aware of the superiority of a posteriori estimation models (Fig. 1). These estimation models are typically being built using data from completed projects, the inputs to a posteriori models have some convictions: they have been measured very accurately, nevertheless, this does not guarantee that the results of the estimation will have the same

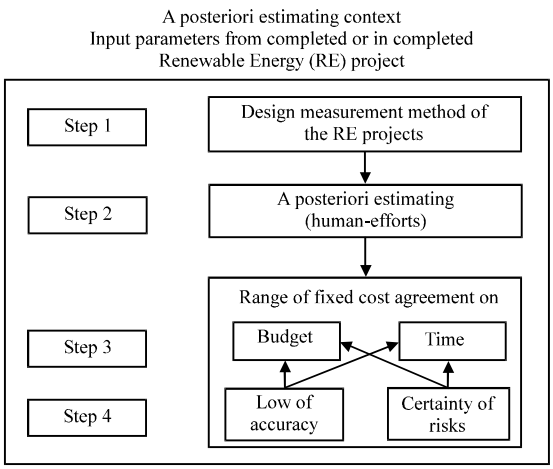

Fig. 1: A posteriori estimation context

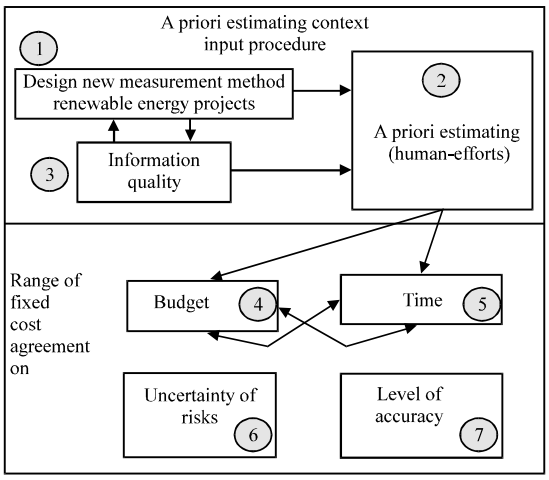

Fig. 2: A priori estimation context

convictions. This is certainly not the distinctive case with the a posteriori models currently available in software engineering.

The presentation of such estimation, using the criteria from academic circles to assess the a posteriori estimation models is quite far from the management expectations for instance, the expectation for a superior estimation model in the literature is that a posteriori model produce estimates that are within the range of ±30% for ±70% of the projects used as inputs to built these models.

Most analyses of existing estimation models and techniques have indicated that most models and techniques do not even meet this low level of quality as expected from the management perspective: such models have been built, using projects already completed that is without unverified and without uncertainty about risks.

Furthermore, a number of estimation models are proposed to the industry without even having been built and verified against past completed projects. Even they include a lot of numbers, most of them are based strictly on unverified and undocumented perceptive deductions, regularly referred to as expert's opinions (Al-Sarayreh and Meridji, 2013).

A priori estimation models: Organizations typically use an estimation approach based either on informal opinion or organizational experience or on a posteriori estimation models developed internally or developed by an outside consultancy (Al-Sarayreh and Meridji, 2013).

For instance, if the estimation model is used very early on the life cycle information is available such as at the feasibility stage, then most of the inputs numbers are based on expectations and are not derived from the application of rigorous measurement procedures, these expectations are definitely numbers but with very little strengths in terms of accuracy, repeatability and reproducibility (Al-Sarayreh and Meridji, 2013).

To speed up the introduction of new products through early cost visibility: the most critical decisions made in a product's life cycle are made in the products development conceptual stages. Using an a priori cost estimation approach can help predict (up to a point) the cost impact made during design decisions, this can have a positive impact on product's boundaries. A priori estimations methods help developers and managers to achieve lower cost product. As well as to avoid cost reduction efforts and using a priori cost estimation in early stages of the project reflects accurate results than using a posteriori cost estimation in the same project (Fig. 2).

Estimation process for a priori model: The estimation process is an important part of the software planning process. It is used to derive the project plan: like process, environment, human resources in the project and project quality. Software cost assessment model estimate the total effort and schedule based on the effort assessment process. A priori estimation process can use the following steps (Al-Sarayreh and Meridji, 2013):

- Classification of the inputs to detailed project to be estimated
- Exploitation of a posteriori models as reproduction models
- Alteration process to take into account variables and information not included in the reproduction process
- Identification ambiguity factors and risk assessment

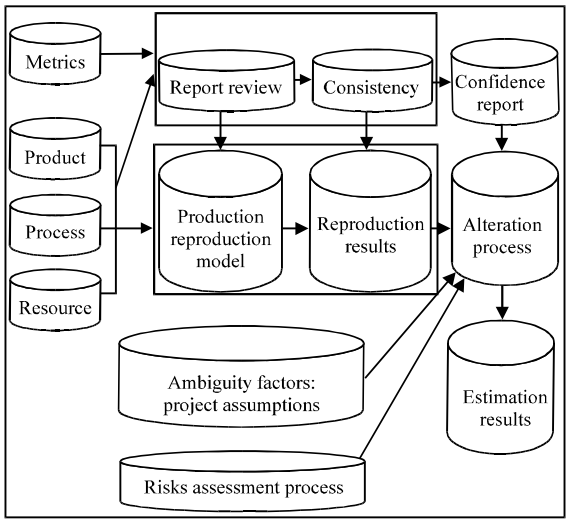

Fig. 3: Estimation process a priori model

- Judgment production at the project and selection levels
- Feedback and enhancement to the a priori estimation process

A high-level view of the estimation process is presented in Fig. 3 with its inputs and output. This estimation process can be further decayed into two major sub-processes: a production reproduction sub-process and a modification of the sub-process.

The production reproduction sub-process includes an efficiency simulation model which takes as input measures of resources, process and products and provides as an output reproduction results. The alteration sub-process takes the reproduction results as input, together with information on ambiguity factors and risk assessment results and it provides as output the results of the full estimation process.

Kitchenham *et al.* (2002) has listed some of the reasons that affected the accuracy of estimations in industry: estimators in industry assume their estimates are poor may be because there have been few empirical studies of actual estimation processes.

One is not aware of any published data that record the contemporary estimates used when projects were undertaken. For example, Hughes investigated how people in industry construct estimates but he did not present any information about how accurate they were.

Empirical studies are usually based on demonstrating the value of some algorithmic estimating method or data-intensive tool (Boehm and Abts, 2000). Thus, it is easy for estimators in industry to believe that if they do not use algorithmic models or data-intensive estimation processes their estimates will be inaccurate.

There are two models to measure the estimate of the accuracy that are popular in the cost estimation community, these models are: The Mean Magnitude Relative Error (MMRE):

$$MMRE = \frac{1}{n\sum_{i=1}^{n}\frac{|Actual\text{-}Estimate|}{Actual}}$$

where, n is the number of projects

The Pred (25) statistics: Pred (25) is the proportion of project estimates within 25% of the actual. For example, if Pred (25) is 0.60, then 60% of the estimates are within 25% of the actual.

Confusion use in a priori and posteriori context: Many estimation models are built and are based on data from past projects, this corresponds to an a posteriori context, though, these models are used classically in a priori context and early in the life cycle. Nevertheless, there is confusion in some published studies as well as some used estimation tools like ONTOCOM, COTS and COCOMO about the concept of a priori and a posteriori contexts, Fig. 4 illustrates the confusion in using a priori throughout a posteriori context.

Furthermore, few independent studies of a number of the a posteriori estimation models have indicated that they typically performed weakly and this in a context where all inputs to these models did not have any ambiguity associated with them.

CHALLENGES OF A PRIORI ESTIMATION CONTEXT

Some of the challenges of a priori estimation context early in the life cycle are listed by (Al-Sarayreh and Meridji, 2013): lacking information at the feasibility stage. Most of the inputs numbers taken are numbers based on prediction with very little strengths in terms of accuracy, they are not derived from the application of thorough measurement procedures. The attained size of the software through an approximation technique from imprecise metaphors of the expected functions of the project at the commencement of the requirements phase. Repeatability and reproducibility will manipulate the quality of the results of the estimation process, numbers will come out of the estimation models.

A posteriori context problems: Some of the problems in a posteriori context are affecting the accuracy of the

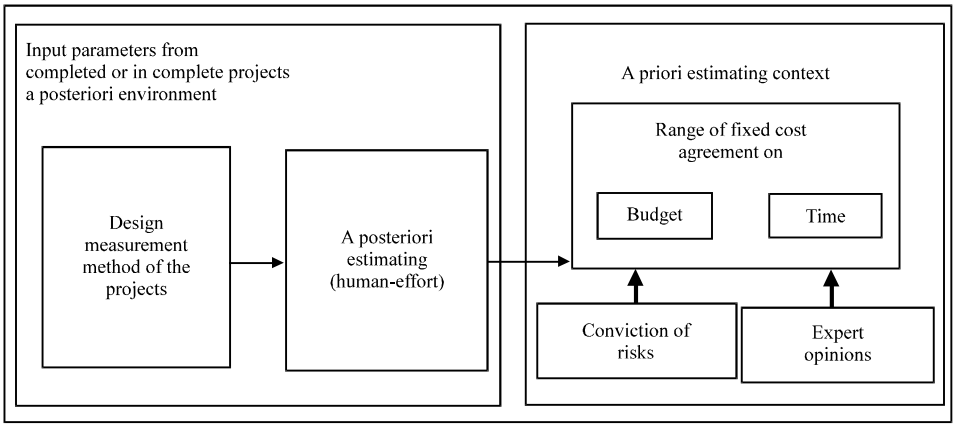

Fig. 4: The confusion use of a priori estimation throughout a posteriori context

estimation models and increasing a decision making risk in the prospect ofprojects development are listed by Al-Sarayreh and Meridji (2013):

The inputs have been measured precisely, this does not promise that the results of the estimation model have the same confidence, this is typical with the posteriori models presently available in software engineering. Most of the estimation techniques and models proposed to practitioners have never been verified separately on past projects. The psychoanalysis of existing estimation models and techniques have indicated that most models and techniques do not even meet this level of quality from the management perspective. Many estimation models are proposed to industry without even having been verified along with precedent accomplished projects, they consist of lots of numbers, most of them based on experts judgments.

A priori and posteriori estimation models: This study presents the most important models used in estimation contexts and techniques.

A priori estimating techniques: The following estimating techniques are considered to be used in a priori context, in addition all of the a priori techniques in this study are non-algorithmic (Al-Sarayreh and Meridji, 2013).

Average method: This method averages two or more of the estimates prepared for the project using other methods. The initial choice of estimation methods is made by the project manager and the independent estimator. Then, some or all of the estimates are averaged, again based on the expertise of the project manager and the independent estimator.

CA-estimacs method: This method is based on a commercial software tool, CA-Estimacs 7.0 that queries the user for project characteristics and applies information from a historical database to develop an estimate. The tool has not been calibrated with the history of the corporate projects, estimates are made based on the database supplied with the tool. The estimate is expressed both in hours and in function points. The input questions vary according to whether the project is client/server, object-oriented, real-time information engineering, maintenance or generic. The independent estimator answers the questions of the tool after consulting with the project manager. The independent estimator helps the project manager to answer the questions consistently.

Comparison method: This method compares the target project to other completed projects that were similar in scope and type. A reference project is chosen and its actual hours are used as a basis for the target project estimate.

Proportion method: This method uses estimates or actual estimates from one or more phases of an existing project. Then, the current estimate is generated by extrapolating to the total development hours using a standard distribution percentage (such as 3-6% for vision and strategy, 12-18% for business systems design and 3-7% for integration).

Widget counting method: This method identifies widgets (repeated characteristics of system development) for the project, counting the number of each and assigning a complexity factor. Past history is used to suggest the number of hours required to produce each widget. The widget estimates are summed. Then, effort for supporting tasks is added to the widget estimate to determine total project hours. Predefined widgets include design, test plans, code, code reviews, unit tests and test reviews.

Expert judgments: Non-algorithmic method, in this method one or more experts are consulted. The experts make estimation based on their own methods and experience. Such as expert-consensus that uses delphi technique to resolve the inconsistency in the estimations (Simperl *et al*., 2006).

Recent models of software cost estimation (a postiriori and non-algorithmic): The following estimation techniques are considered to be used in a posteriori context in addition all of the a priori techniques in this study are non-algorithmic (Al-Sarayreh and Meridji, 2013).

Analogy costing: This methodis used when there areprojects in the same application that have been completed. The estimation of cost for the new project is done by analogy with the finished projects.

Parkinson's law: The cost is defined by using available resources rather than by using an objective evaluation.

Pricing to win: The estimation of cost is based on available budget. The estimation of the effort is dependent on the customer's budget and not on the project functionality. The project cost is decided based on a proposal outline and the development is constrained by that cost.

Top-down: The estimation begins at the system level and evaluates the overall system functionalities. This method is used without having any knowledge about the system architecture and their components.

Bottom-up: This method begins making estimations at a component level and the estimation of the effort is made for each component.

Recent models of software cost estimation (a postiriori and algorithmic): The following estimating techniques are considered to be used in a priori context; in addition all of the a priori techniques in this study are algorithmic (Table 1).

The strengths and the weaknesses of a posteriori cost estimation models: This study presents the major cost estimation models with their strengths and weaknesses in Table 2 and 3.

Table 1: Cost estimation models: A posteriori and algorithmic

Measure of software size	A posteriori/Algorithmic model	Efforts
Line of Code (LOC)	Walston-Felix Model Produced by IBM-FSD Model, 1977 Used by IBM to estimate programs Some statistical concerns	$E = 5.2(KLOC)^{0.91}$
	Bailey-Basili Model Produced by Bailey-Basili in 1981 Statistical analysis of factors and size	$E = 5.5+0.73(KLOC)^{1.16}$
	Boeing Model (simple) Produced by Black *et al*. in 1977 Similar to COCOMO but simpler Out of use Poor estimates	$E = 3.2(KLOC)^{1.05}$

Table 2: Cost estimation models: a posteriori and algorithmic (Contd.)

Measure of software size	A posteriori/Algorithmic model	Efforts
Line of Code (LOC)	Doty Model for KLOC>9 Produced by Herd in 1977 Extended the SDC Model Problems with stability	E = 5.288 (KLOC) 1.047
Function Point (FP)	Albrecht and Gaffney Model	E = -13.39+0.0545 (FP)
	Kemerer Model	E = 60.62+7.728 (FP)
	Matson, Barnett and Mellichamp Model	E = 585.7+15.12 (FP)

Table 3: Cost estimation models: strengths and weaknesses (Dillibabu and Krishnaiah, 2005; Al-Sarayreh and Meridji, 2013)

Models/Strengths	Weaknesses
Algorithmic Model	
Objective, repeatable results, analogy formula, efficient, good for sensitivity analysis and objectively calibrated to experience	Subjective inputs, calibrated to past, not to the future and assessment of exceptional circumstances, algorithms are suitable for specificsoftware development
Expert judgment	
Assessment of representativeness, interactions, exceptional circumstances, relatively cheap estimation method. It can be accurate if experts have direct experience of of similar systems	No better than the expertise of the individual participants, biases and incomplete recall. Very inaccurate if there are no experts, sometimes questionable may not be consistent

Table 3: Continue

Models/Strengths	Weaknesses
Analogy	
Based on representative experience, accurate if project are data available	Impossible if no comparable project has been tackled Needs systematically maintained cost database. Similar projects may not exist, historical data may not be accurate
Parkinson's Law	
Correlates to some experience, often win the contract	May reinforce poor practices, system is usually unfinished
Pricing to win	
Often gets the contract	Generally produces large overruns
Top-down, Albrecht and Gaffney	
System level focused and efficient, require minimal project details, faster and easier than bottom-up method	Less detailed basis. perform the estimate early in the life cycle
Bottom-up, Albrecht and Gaffney	
More detailed basis and more stable	May overlook system level cost and requires more effort Difficult to perform the estimate early in the life cycle

CONCLUSION

Software cost modeling in software engineering was initiated with the "Software Development Corporation (SDC) study of 104 attributes of 169 software projects. This led to some useful partial models in the late 1960's and early 1970's".

The late 1970s was a high point of new models such as SLIM, Checkpoint, PRICE-S, SEER and COCOMO. The majority of these researchers started working at the same time on developing models of cost estimation. They all faced the same problems like: software grew in size and in complexity, making it very difficult to estimate accurately the cost of software development.

Cost estimation models can be categorized into the following: algorithmic and non-algorithmic, each one of them having strengths and weaknesses but the accuracy is a key aspect for selecting a cost estimation model.

The rapid changing of software development made it very difficult to develop parametric models that give high accuracy for software development in all domains. The cost of software development keep on raising and practitioners are continually concerned about having accurate predictions.

One of the important objectives of the software engineering community has been the development of models that usefully explain the development life cycle and accurately predicts the cost of developing a software product.

Estimators in industry assume their estimates are poorly done. This may be due to few empirical studies of actual estimation processes. People in industry construct estimates without presenting any information about the accuracy of these estimates.

One is not aware of any published data that record the contemporary estimates used when projects were undertaken.

Many researchers used different terminologies in cost estimation field, however, there is a lack of standardized terminology and it is difficult to identify the meaning, since, many use "prediction" instead of "estimation" and "maintenance task" instead of, e.g., "software development", "function points" instead of more general estimation terms, a variety of terms are used instead of "software", e.g., "system", "maintenance", "project" and "task". "A priori" and "a posteriori" terms are used in other point of view to distinguish between two different types of knowledge justification and argument: a priori knowledge is known without having any experience and a posteriori knowledge is confirmed throughout experience.

Many estimation models are proposed to industry without even having them built and verified against past completed projects, they include lots of numbers, most of them based on unverified and undocumented subjective guesses, often referred to as "expert's opinions".

ACKNOWLEDGMENT

The researchers would like to thank the University of Jordan, The Hashemite University, and University of Petra, for their support to this research project.

REFERENCES

Al-Sarayreh, K.T. and K. Meridji, 2013. Towards apriori and posteriori estimation models for renewable energy projects in software engineering. Proceedings of the 2013 1st International Conference and Exhibition on the Applications of Information Technology to Renewable Energy Processes and Systems, May 29-31, 2013, IEEE, Amman, Jordan, ISBN:978-1-4799-0713-7, pp: 101-106.

Boehm, B.W. and C. Abts, 2000. Software development cost estimation approaches: A survey. Ann. Software Eng., 10: 177-205.

Cuadrado-Gallego, J.J., L. Fernandez-Sanz and M.A. Sicilia, 2006. Enhancing input value selection in parametric software cost estimation models through second level cost drivers. Software Qual. J., 14: 339-357.

Dillibabu, R. and K. Krishnaiah, 2005. Cost estimation of a software product using COCOMO II. 2000 model-a case study. Int. J. Project Manage., 23: 297-307.

Grimstad, S. and M. Jorgensen, 2006. A framework for the analysis of software cost estimation accuracy. Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, September 21-22, 2006, ACM, New York, USA., pp: 58-65.

Kitchenham, B., S.L. Pfleeger, B. McColl and S. Eagan, 2002. An empirical study of maintenance and development estimation accuracy. J. Syst. Software, 64: 57-77.

Mendes, E., C. Lokan, R. Harrison and C. Triggs, 2005. A replicated comparison of cross-company and within-company effort estimation models using the ISBSG database. Proceedings of the 11th IEEE International Symposium on Software Metrics (METRICS'05), September, 19-22, 2005, IEEE, Como, Italy, pp: 10-36.

Mohagheghi, P., B. Anda and R. Conradi, 2005. Effort estimation of use cases for incremental large-scale software development. Proceedings of the 27th International Conference on Software Engineering, May 15-21, 2005, Norway, pp: 303-311.

Nelson, E.A., 1967. Management Handbook for the Estimation of Computer Programming Costs. System Development Corporation, Santa Monica, California, USA., Pages: 141.

Putnam, L.H. and W. Myers, 1991. Measures for Excellence: Reliable Software on Time, within Budget. 1st Edn., Prentice Hall, Upper Saddle River, New Jersey, USA.,.

Putnam, L.H., 1978. A general empirical solution to the macro software sizing and estimating problem. IEEE Trans. Software Eng., 4: 345-361.

Simperl, E.P.B., C. Tempich and Y. Sure, 2006. Ontocom: A cost estimation model for ontology engineering. Proceedings of the 5th International Conference on Semantic Web, November 5-9, 2006, Springer, Berlin, Heidelberg, pp: 625-639.

Song, T.H., K.A. Yoon and D.H. Bae, 2007. An approach to probabilistic effort estimation for military avionics software maintenance by considering structural characteristics. Proceedings of the 14th International Conference on Asia-Pacific Software Engineering (APSEC'07), December 4-7, 2007, IEEE, Aichi, Japan, pp: 406-413.

Stellman, A. and J. Greene, 2005. Applied Software Project Management. O'Reilly Media, Sebastopol, California, USA., Pages: 303.

Summerville, I., 2004. Software Engineering. 7th Edn., Pearson Addison Wesley, Boston, Massachusetts, USA., ISBN:9780321210265, Pages: 759.

Wieczorek, I., 2002. Improved software cost estimation-A robust and interpretable modelling method and a comprehensive empirical investigation. Empirical Software Eng., 7: 177-180.