Towards Apriori and Posteriori Estimation Models for Renewable Energy Projects in Software Engineering

Khalid T. Al-Sarayreh
Department of Software Engineering
Hashemite University
Zarqa 13115, Jordan

Kenza Meridji
Department of Software Engineering
University of Petra
Amman 11196, Jordan

***Abstract*— currently, cost estimation for renewable energy projects is one of the important issues in software engineering projects. It has been observed in the industry that cost estimates of renewable energy projects often differ from the final costs by a factor of two or more; such large overestimates impact on process integrity and, ultimately, on final quality. RE development organizations are expected to deliver projects on time and on budget in a context of fixed-price contracts. To meet these market demands, managers must determine well in advance such estimates which called is the *apriori cost* estimation. These organizations typically use an estimation techniques based either on informal personal or organizational experience, or on *aposteriori* estimation models developed internally or developed by outside consultancy. This paper collects most of the common used models of the cost estimation models throughout the apriori and aposteriori contexts, and we highlights some of their mixed uses between these estimating models in software engineering development projects for RE and lists some of the estimating techniques used in A priori and A posteriori contexts.**

Keywords: a priori context; a posteriori context; cost estimation models; renewable energy.

I. Introduction

Cost estimation is the process of predicting the effort required to develop a system; accurate cost estimates are critical to both developers and customers. The size measure of RE development depends on the human effort, and most cost estimation methods focus on this aspect and give estimates in terms of person-months or hours. Most cost models are based on the size measure and the accuracy of size estimation directly affects the accuracy of cost estimation. Most cost estimation models attempt to produce an effort estimate, which can then be converted into the project duration and cost.

Cost Estimation is defined as an approximation of the probable total cost of a product, program, or project, computed on the basis of available information. Cost Estimation Models can be classified into algorithmic and non-algorithmic, each has its own strengths and weaknesses, but a key aspect in selecting a cost estimation model is the accuracy of its estimates.

Software project estimation models are challenge to most organizations and to their customers who suffers from RE development projects significantly over budget, with significant delays in schedules, less functionality than guaranteed and with unknown levels of quality. Estimation and quality are two of the most common issues facing managers and practitioners as well as their users and organizations.

Fundamentally, there are two types of cost estimation contexts in software engineering: apriori and aposteriori contexts. The terms apriori and aposteriori are used in philosophy to distinguish two different types of knowledge, justification, or argument: apriori knowledge is known independently of experience, and a posteriori knowledge is proven through experience.

Many estimation models are built and based on data from past projects; this corresponds to an aposteriori context; however, these models are used typically in an apriori context and early in the development life cycle. Typically in software engineering, the aposteriori models would not have been verified in an apriori context. There is confusion in some published studies as well as some used estimating tools like ONTOCOM, COTS, and COCOMO about the concept of apriori and aposteriori contexts.

Some of the reasons behind these problems are:

1) Many estimation models are built and based on data from past projects: this corresponds to an 'a posteriori' context; however, these models are typically used in an 'a priori' context, and fairly early in the development life cycle. Typically, the a posteriori models would not have been verified in an a priori context.

2) Most cost estimation approaches have little statistical basis and have not been validated.

3) The reliability of inputs to cost estimation models varies widely; and this could reduce the validity of historical data as a basis for validation of these models.

There are no studies differentiate between the use of the a priori and aposteriori estimation context in software engineering for development the renewable energy projects.

This paper is organized as follows. Section II presents the related work. Section III presents the types of estimation models. Section IV presents the confusion use of a priori and a posteriori context. Section V presents the estimating techniques, methods and models in a priori and a posteriori view. Finally, a conclusion is presented in Section VI.

II. Related Work

In the software engineering literature, there is a lot of work claiming to address early estimation, but a closer look at these papers indicates that they are in practice using an aposteriori context as an apriori context. On one hand, there are many early estimation methods could be used in a priori contexts such as the ones listed by Kitchenham [1] for example: Average, CA-Estimacs, Comparison, Proportion, Widget counting and Delphi methods. While Nelson [2] introduced early estimating model; could be used in a *priori* context.

On the other hand, there are many other estimating techniques and models constructed and developed from information available after the completion of projects [3] such models should be qualified as *a posteriori* estimation models, such as COCOMO [4], SLIM [5], Checkpoint [6] , PRICE-S [7], SEER [8], Walston-Felix Model [9], Bailey-Basili Model [10], Boeing Model [11], Doty Model for KLOC [12], Albrecht and Gaffney Model [13], Kemmerer Model [14], Matson, Barnett & Mellichamp Model [15].

Cost Modeling in software engineering was initiated with the Software Development Corporation (SDC) study of 104 attributes of 169 software projects [16]. 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 followed a priori context in software engineering, for example Kitchenham [1]. 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 [17], the authors introduced a priori cost model, but move on to a posteriori approach to complete their estimation.

III. Types of Estimation Models

Estimation models are essential for effective software engineering for RE project and their management. During the past three decades, many cost estimation techniques have been proposed to predict the cost. A common weakness of most models is their limited usefulness to predict the cost accurately at an early stage of the development life cycle. To understand this problem, there are fundamentally two types of estimation models: A posteriori estimation models and *A priori* estimation models.

A. A posteriori estimation models

Estimation models are still immature domain of knowledge in software engineering where most of the technical estimation techniques and models proposed to practitioners have never been verified independently on past projects; and for many of the models that have been built with past projects, most still have a low degree of accuracy in their estimation.

Software engineers as well as managers who are using a posteriori estimation models should also be aware of the quality of these a posteriori estimation models Fig. 1. These estimation models are typically being built using data from completed projects; the inputs to these a posteriori models have some certainties: they have been measured very accurately; however, this does not guarantee that the outcomes of the estimation will have the same certainty. This is certainly not the typical case with the a posteriori models currently available in software engineering.

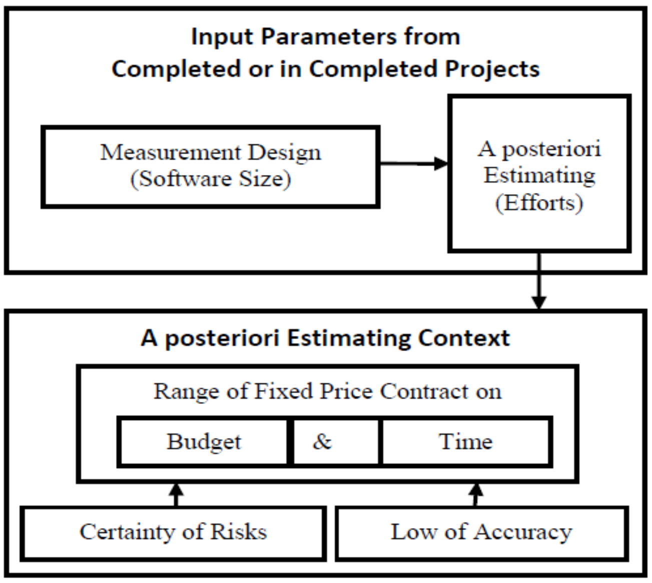

Fig. 1. Typical context of a posteriori estimation model

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 an a posteriori model produce estimates that are within the range of ±30% percent for ±70% percent of the projects used as inputs to built these models.

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

Furthermore, a number of estimation models are proposed to the industry without even have 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 experts' opinions.

B. A priori estimation models

Organizations typically use an estimation approach based either on informal personal or organizational experience, or on *a posteriori* estimation model developed internally, or developed by an outside consultancy.

For instance, if the estimation model is used very early on the life cycle when only insufficient information is available such as at the feasibility stage, then most of the input numbers are based on expectation and are not derived from the application of rigorous measurement procedures; these expectation are definitely numbers, but with very little strengths in terms of accuracy, repeatability and reproducibility.

To speed the introduction of new product through early cost visibility: some of the most critical decisions made in a product's life are those made during the conceptual stages of product development. Using an a priori cost estimation approach can help predict (up to a point) the cost impact of design decisions early in the process which, successively, can have a positive impact on product margins. Apriori has proven that it can help us achieve lower cost product designs and avoid post-launch 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 see Fig. 2.

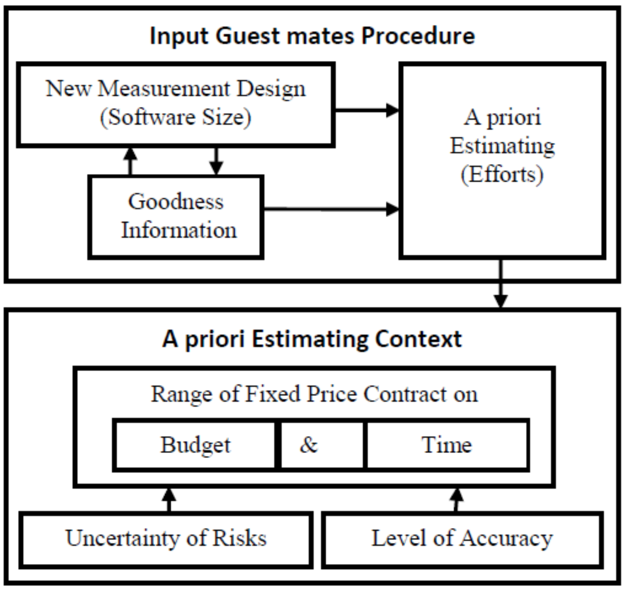
Input Guest mates Procedure
New Measurement Design (Software Size)
Goodness Information
A priori Estimating (Efforts)
A priori Estimating Context
Range of Fixed Price Contract on
Budget
&
Time
Uncertainty of Risks
Level of Accuracy

Fig. 2. Typical context of apriori estimation model.

IV. Confusion use of Apriori and Posteriori Context

Many estimation models are built and based on data from past projects; this corresponds to an a posteriori context; however, these models are used typically in an *a priori* context and early in the development life cycle. However, there is confusion in some published studies as well as some used estimating tools like ONTOCOM, COTS, and COCOMO about the concept of a priori and a posteriori contexts, Fig. 3 illustrates the confusion use of 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 poorly, and this in a context where all inputs to these models did not have any uncertainty associated with them.

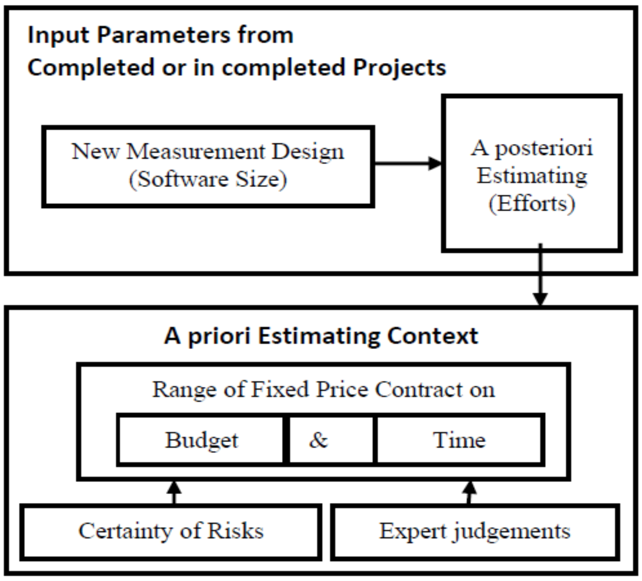
Input Parameters from Completed or in completed Projects
New Measurement Design (Software Size)
A posteriori Estimating (Efforts)
A priori Estimating Context
Range of Fixed Price Contract on
Budget
&
Time
Certainty of Risks
Expert judgements

Fig. 3. The confusion use of c Context.

A. challenges of a priori estimation context

Some of the challenges of a priori estimation context early in the life cycle are:

1) Insufficient information at the feasibility stage.
2) Most of the input numbers are predict numbers with very little strengths in terms of precision; they are not derived from the application of thorough measurement procedures.
3) The obtained size of the software through an approximation technique from imprecise description of the expected functions of the project at the beginning of the requirement phase.
4) Repeatability and reproducibility will influence the quality of the outcomes of the estimation process; numbers will come out of the estimation models.

B. A posteriori context problems

Some of the problems in a posteriori context that affecting the accuracy of the estimation models and increasing a decision making risk in the development projects in the future:

1) The inputs have been measured precisely, this does not guarantee that the outcomes of the estimation model have the same certainty; this is typical with the posteriori models currently available in software engineering.
2) Most of the estimation techniques and models proposed to practitioners have never been verified independently on past projects.
3) The analyses of existing estimation models and techniques have indicated that most such models and techniques do not even meet this level of quality as perceived from the management perspective.
4) Many estimation models are proposed to industry without even have been verified against past completed projects; they include lots of numbers, most of them based on experts opinions.

V. A PRIORI AND A POSTERIORI ESTIMATING

This section presents the most important models used in estimation contexts and techniques.

A. 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 paper are non-algorithmic.

1) Average Method: This method averages two or more of the estimates prepared for the project using the other methods. The initial choice of estimating 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
2) 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.
3) 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.
4) Proportion Method: This method uses estimates or actual 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).
5) 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.
6) Expert Judgments: Non-algorithmic method; this method involves consulting one or more experts. The experts provide estimates using their own methods and experience. Expert-consensus mechanisms such as Delphi technique will be used to resolve the inconsistency in the estimates [17].

B. Recent Models of software Cost Estimation (A postiriori and Non-Algorithmic)

1) Analogy Costing: this technique is applicable when other projects in the same application domain have been completed. The cost of a new project is estimated by analogy with these completed projects.
2) Parkinson's Law: the work expands to fill the time available. The cost is determined by available resources rather than by objective assessment. If the software has to be delivered in 12 months and five people are available, the effort required is estimated to be 60 person months.
3) Pricing to win: the software cost is estimated to be whatever the customer has available to spend on the project; the estimated effort depends on the customer's budget and not on the project functionality. However, when detailed information is lacking it may be the only appropriate strategy: the project cost is agreed on the basis of an outline proposal and the development is constrained by that cost; a detailed specification may be negotiated or an evolutionary approach used for system development.
4) Top-down: start at the system level and assess the overall system functionality and how this is delivered through sub-systems, Usable without knowledge of the system architecture and the components that might be part of the system. Takes into account costs such as integration, configuration management and documentation. It can underestimate the cost of solving difficult low-level technical problems.
5) Bottom-up: Start at the component level and estimate the effort required for each component. Add these efforts to reach a final estimate, Usable when the architecture of the system is known and components identified; this can be an accurate method if the system has been designed in detail; it may underestimate the costs of system level activities such as integration and documentation.

C. The Strengths and the Weaknesses of a posteriori Cost Estimation Model

This section presented the major cost estimation models with their strengths and weaknesses, which are summarized in Table I.

VI. 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 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.

Cost estimation models can be classified into algorithmic and non-algorithmic, each with their own strengths and weaknesses, but a key aspect in selecting a cost estimation model is the accuracy of its estimates.

The rapid changing of software development made it very difficult to develop parametric models that give high accuracy for software development in all domains. Software development costs continue to increase and practitioners continually express their concerns over their lack of ability to predict accurately the costs involved.

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

Estimators in industry assume their estimates are poor may be because there have been few empirical studies of actual estimation processes and people in industry construct estimates, but he did not present any information about how accurate they were

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

Many researchers used different terminology in cost estimation field; there is a lack of standardized terminology, and 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 used instead of "software", e.g., "system", "maintenance", "project", and, "task"

The terms "a priori" and "a posteriori" are used in philosophy to distinguish two different types of knowledge, justification, or argument: a priori knowledge is known independently of experience, and a posteriori knowledge is proven through experience.

Many estimation models are proposed to industry without even have been 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.

Table I: Cost estimation models: strengths and weaknesses [18].

Models	Strengths	Weaknesses
Algorithmic Model	Objective, repeatable results, analogy formula, efficient, good for sensitivity analysis & objectively calibrated to experience	Subjective inputs, calibrated to past, not to the future & assessment of exceptional circumstances, Algorithms are suitable for specific software development
Expert Judgment	Assessment of representativeness, interactions, exceptional circumstances, Relatively cheap estimation method. Can be accurate if experts have direct experience of similar systems	No better than the expertise of the individual participants, biases & incomplete recall. Very inaccurate if there are no experts, sometimes questionable; may not be consistent.
Analogy	Based on representative experience, accurate if project 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 focus and efficient, Require minimal project detail, 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 & requires more effort. Difficult to perform the estimate early in the life cycle

REFERENCES

[1] Kitchenham, B., S. Lawrence, et al. (2002). "An empirical study of maintenance and development estimation accuracy." The Journal of Systems and Software, Elsevier Science Inc, 2002 64: 57-77.

[2] Nelson, E. "Management Handbook for the Estimation of Computer Programming Costs." Systems Development Corporation, 1966.

[3] Grimstad, S. and M. Rgensen, "A framework for the analysis of software cost estimation accuracy." Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, 2006.

[4] Putnam, L. H. and W. Myers, "Measures for Excellence: Reliable Software on Time, within Budget." Prentice Hall Professional Technical, 1991.

[5] Jensen, R., "An improved macrolevel software development resource estimation model." 5th ISPA Conference April 1983: 88-92, 1997.

[6] Putnam, L. H., "A general empirical solution to the macro software sizing and estimating problem." IEEE Transactions on Software Engineering: 354-361, 1978.

[7] Park, R., "The Central Equations of the PRICE Software Cost Model, 4th COCOMO Users', Group Meeting." 1988.

[8] Juan Cuadrado Gallego, Luis Fernandez Sanz, et al., "Enhancing input value selection in parametric software cost estimation models through second level cost drivers" Software Quality Journal, Springer Netherlands 14(4), 2006.

[9] Song, T. H., K. A. Yoon, et al., "An approach to probabilistic effort estimation for military avionics software maintenance by considering structural characteristics." Asia-Pacific Software Engineering Conference: 406-413, 2007.

[10] Stellman, A. and J. Greene, "Applied Software Project Management." Cambridge, MA: O'Reilly Media, 2005.

[11] Sommerville, I., "Software Engineering." Pearson Addison Wesley 7th edition (Chapter 26), 2004.

[12] Abts, C., B. W. Boehm, et al., "COCOTS: A cots software integration lifecycle cost model - model overview and preliminary data collection findings, 2000.

[13] Mendes, E., C. Lokan, et al., "A replicated comparison of cross-company and within-company effort estimation models using the ISBSG database." 11th IEEE International Software Metrics Symposium: 36, 2005.

[14] Mohagheghi, P., B. Anda, et al., ""Empirical software engineering: Effort estimation of use cases for incremental large-scale software development"." Proceedings of the 27th International Conference on Software Engineering, 2005.

[15] Wieczorek, I., "Improved Software Cost Estimation. A Robust and Interpretable Modelling Method and a Comprehensive Empirical Investigation." Ph.D. Theses in Experimental Software Engineering 7(Fraunhofer IRB Verlag), 2001.

[16] Boehm, B. and C. Abts, "Software Development Cost Estimation Approaches – A Survey1." University of Southern California, 2000.

[17] Simperl1, E. P. B., C. Tempich2, et al., "ONTOCOM: A Cost Estimation Model for Ontology Engineering", 2005.

[18] Dillibabu, R. and K. Krishnaiah, "Cost estimation of a software product using COCOMO II.2000 model – a case study." International Journal of Project Management, Elsevier: 297-307, 2005.