# ORGANIZATIONAL SOFTWARE COST ESTIMATION MODEL BASED ON HISTORICAL DATA

**Abdulhameed Alelaiwi**
Department of Software Engineering, College of Computer & Information Sciences,
KingSaudUniversity, P.O. Box 51178, Riyadh11543,
Kingdom of Saudi Arabia

*ABSTRACT: Accurate software development estimation continues to be a difficult problem for government and industry. Accuracy is determined by measuring the difference between the estimated cost and the actual cost. The literature contains abundant documentation of automation projects exceeding original costs and schedule estimates. Unexpected budget increases and schedule delays reduce sponsor confidence in the original software development estimates and damage the reputation of the estimation process.*
*In this paper, authors are involved in conceptualizing a model that will overcome some of the limitations cited in this paper. The alternative technique is to develop a model unique to a particular environment, which should more accurately reflect that environment. A number of studies suggest that local models are more accurate than all-purpose models. Studies have shown that most organizations that have built up a considerable database over time, generally have a history of software management, and would have a custom model in place as opposed to deploying a generic one.*

**Key words:** *Software Cost Estimation (SCE), Algorithmic Models, LOC, Markov Chain, Data Collection, AHP.*

## 1. INTRODUCTION

To date, most work carried out in the software cost estimation field has focused on algorithmic cost modeling. In this model, costs are analyzed using mathematical formulas linking costs or inputs with metrics to produce an estimated output. The formulas used in a formal model arise from the analysis of historical data. In general algorithmic models estimate project effort using a process where a formula is modified according to project characteristics. Model accuracy and consistency depend on the ability of the size metrics in a database to represent the product, process, and environment of prior development projects. Parametric models are calibrated for each development environment using data from project metric database.

A key finding of the survey of existing software cost estimation models is that most of the models are limited in nature. These limitations emanate from three fundamental aspects of the estimation process:

1-Some models use historical data obtained from specific development environment. Environments do change from project to project. Equations and methods from these models of specific environments give wrong estimates for differing environments.

2-Models that estimate the overall cost of a software project omit details that play a significant role in providing accurate input. On the other hand, models that estimate costs based on predefined phases can be erroneous because the new project for which the model is used may have a different set of phases.

3-In general, estimation is accomplished during initial phases of a project. For example, estimates are customarily made during the feasibility and requirements phases. This estimation in early stages is less accurate because of incomplete information, especially if the project has new innovative steps.

Literature points out this weakness and declares that most cost estimation models perform poorly because "they were developed from analysis of available legacy data sets." Literature states that the models do not have the mathematical rigor needed to accommodate the lack of stationary in the statistics that are being measured, which necessitates continually adjusting constants that are used to generate the effort size equation to fit every data set.

Some models rely heavily on delivered code size (LOC) as an essential parameter. LOC has been known to be very misleading. The validation of such models based on fuzzy independent variables is not a precise task, and it is usually difficult to arrive at a consensus on the manner in which subjective cost drivers ought to be measured.

1) Generic models have the following limitations:

2) The size/effort relationship can be inconsistent over various environments.

3) Generic models are calibrated to a specific profile of projects and perform best when it is applied to a project of that specific profile.

Generic algorithms can cause inaccurate predictions unless calibrated across environments and as conditions change. Thus, generic models are not transferable between organizations

The estimates are usually calculated only during the early stage of a project.

Some cost models estimate the overall cost of a project; others estimate the effort required in a predefined phases. Estimating the overall cost of the projects lacks necessary details, while estimating cost in predefined phases forces the use of theses phases.

- Estimation process is usually done once during the early stage of the project.
- An estimators needs practice at developing estimates.

The models published in the literature are not portable to other development environments in their uncelebrated form. The models must be calibrated for each development environment. There is no formal way in the literature to do the calibration. Uncelebrated models inaccurately compute the effort required for tasks because organizations have different processes, products, development environment, and review requirements. For example Kermer in [1] used a single set of projects from an anonymous organization to check the accuracy and consistency of four popular models. The results indicate that there is a wide variation in the results from the models.

In this paper, authors are involved in devising new models that will overcome these limitations or at least they can lift some of the limitations. This paper documents the findings of the survey with the hope to enable researchers in correcting estimation problems in their model development effort.

Conventional cost models are sometimes based on parameters that do not truly reflect the real nature of the programming effort, as they rely heavily on historical data to arrive at estimates. For this reason, it is time-consuming to refine the model to suit one's needs and becomes a matter of trial-and-error.

## 2. LITERATURE REVIEW

Estimators always feel a challenge to estimate the cost of software, because non-estimation usually results in increased budget and schedule. Researchers are working to solve this issue and hence some related work is presented below.

A morphological-rank-linear system was presented by Ricardo el at. This system is used to solve the software cost estimation problem.MRL is based on the relationship between Morphological –rank operator and Finite impulse response. This system is a linear combination between MR and FIR. For MR training algorithm was used and for MRL two metrics were used to solve the issues. To check the performance of a proposed MRL model, evaluation function method was designed which showed that MRL model is more consistent.MRL has both linear and non-linear components and is simple in its approaches. This makes it more applicable then the other one [2].

Nikoloas and Lefteris introduced a virtual tool which is Regression Error Characteristics (REC) analysis, another method to estimate the software cost. This method involves geometrical properties along with simple inspection graphs to compare and validate different proposed models.REC uses cumulative distribution function for the measurement of prediction error. This method provides better management of projects as well as helps out to find the effects of errors by identifying the types of errors that affects the software cost [3].

Research has shown that in order to calculate the software cost, sometimes parametric models are used instead of universal mathematical expression .In this regard, Javier et al. presented a segmented model based on fuzzy clusters. In fuzzy cluster different mathematical models are used to calculate software cost in a particular time for estimation purpose [4].

Another popular technique for software development cost is ESTIMATION BY ANALOGY (EBA).It is a complicated process. . Nikolaos *et al* [3] worked on the possible improvement of this method. The technique works on bootstrap method. They increased the efficiency of this method by investigating aptitude of iterated bagging to overcome the estimation error of EBA. They gave a trial to this method and found that the system greatly reduced the errors and increased the efficiency of EBA [5].

Another method to reduce the software development cost was proposed by Ricardo et al. He designed morphological-rank linear (MRL) model. In MRL, a modified genetic algorithm (MGA) was used to increase the efficiency of software cost estimation. Several experiments were done to test this method based on 6 databases of software products and the results were compared with machine learning models. It is a simple and easy method which contains linear and non linear components [6].

Another model based on the same issue was proposed by Vinay Kumar *et al.* In this method he used WAVELETE NEURAL NETWORK (WNN).The technique used for this method was Training Algorithm (TAWNN). WNN used two types of transfer function which are Morlet function and Gaussain function. Analysis was done and the results showed that both WNN and TAWNN were reliable methods to estimate the software cost [7].

Another technique such as constructive cost model (COCOMO) was also used to estimate the software cost. The technique is based on three stages. Stage one which is the basic level is based on low accuracy which makes quick and rough estimation. Stage two which is the intermediate level considers more cost factors. Third level which is the detailed level is based on individual project level [8]. Qin and Fang discussed three most popular methods that are being used for software cost estimation such as Scientists determined that the method, Analogy method and parameter model method. In Parameter model COCOMO is most widely used because of its estimation equations that provide flexible and reliable input and output factors for scheduling and work load during the project development [9].

Different data mining techniques were introduced by Zeynab and Farhad. To increase the efficiency of software cost estimation they tested their technique by NASA'S PROJECT.COCOMO succeeded in improving SCE. They also proved that artificial intelligence methods are more reliable then algorithmic methods. On comparing Support Vector Regression (SVR), K-Nearest Neighbors (KNN), Linear Regression (LR) and Artificial Neural Network (ANN), they found that ANN and SVR are more efficient methods [10].

Ricardo et al. proposed another morphological approach based on hybrid artificial neuron. This approach consists of two factors which are mathematical morphology and algebraic foundation both based on complete lattice theory. Both methods were tested by DILATION –EROSION PERCEPTRON (DEP) and the results indicated the efficiency of this process in all regards [11].

## 3. THE STOCHASTIC MODEL

The approach to the development of the stochastic model involves a number of activities that need to be performed. The generation of the proposed model involves the following features/activities:

- Collection and maintenance of a database of past projects.
- Statistical analysis of the data to derive an appropriate model relationship.

- Identification of completed projects with properties similar to the current project and their costs used as basis for estimating the current projects.
- Calibration of models using only local data, and feedback data as the project progresses.
- Use of any set of software artifacts desired.
- Use of any lifecycle models like sequential model, or spiral model.
- Capability to repeat the estimation process as needed throughout the duration of a project.

For estimating cost on a new project, the proposed model will use the data of similar projects that were previously completed. The data will include project phases, artifact applied, time durations, effort, cost and other key information. The model will use the data on completed projects to construct a Markov Chain along with state probabilities to estimate cost of the new project.

## 4. COLLECTION OF THE PROJECTS DATA

All well known software estimation models are derived using empirical techniques and this paper considers data drawn from a specific organization

It is necessary to have more rigorous approach to data collection. The data collection approach should contain information of the product and software artifacts used. Metrics is quantitative means of recording the history of a product. The proposed model assumes an organization has useful techniques for gathering and applying metric information to cost estimation. The data gathered should be in a form useful to the estimator to build and update the model. The data should include both product metrics which describe the features of the system built, and process metrics which describes features of the software artifacts used to build the system. It is hard to identify specific processes that should be recorded and used by every organization; each organization is unique. However, metrics recorded for the purpose of improving cot estimation should be able to satisfy the following:

- Actual effort of the system development at any stage of the various software artifacts must be recorded.
- All estimates and re-estimates are recorded. To determine the accuracy of estimates, a complete record of all estimates must be maintained.

The characteristics of the completed products include the size measured in some suitable units (e.g., Source Line of Code, Function Points), a description of the functionality of the system, classification of type of software, and any other information that characterizes the product.

The approach to the development of the new model involves a number of activities that need to be performed. The generation of the proposed model involves the following features/activities:

- Collection and maintenance of a database of past projects.
- Statistical analysis of the data to derive an appropriate model relationship.
- Identification of completed projects with properties similar to the current project and their costs used as basis for estimating the current projects.
- Calibration of models using only local data, and feedback data as the project progresses.
- Use of any set of software artifacts desired.
- Use of any lifecycle models like sequential model, or spiral model.
- Capability to repeat the estimation process as needed throughout the duration of a project.

This paper conceptualizes a model that will overcome some of the limitations cited in [12]. The new technique is to develop a model unique to a particular environment (or an organization), which should more accurately reflect that environment. A number of studies suggest that local models are more accurate than all-purpose models. Studies have shown that most organizations that have built up considerable databases over time, generally have a history of software management, and would have a custom model in place as opposed to deploying a generic one.

The approach to the development of the new model involves a number of activities that need to be performed. Data gathering is an essential part to build the model.

The model will improve the cost estimation process by gathering related data on previous projects. The simplest way to gather data is to have a stable workforce so the project and process data are maintained in the memory of individuals. However, relying on individual's imperfect memory is completely inadequate for software project. To overcome such limitations, it is necessary to have more rigorous approach to data collection. The data collection approach should contain information of the product and software artifacts used in each phase of software process. Software Metrics is quantitative technique of recording the history of a product. The model will have useful technique for gathering and applying metrics information to cost estimation. (Software Engineering Institute, CMU, IEEE forms will be used).

The data represent automated system projects that are completed and deployed. The data consists of function of time f(t), (Pi), APSAi, and Cost. f(t) represents the time to traverse from state to state in the Markov Model. Pi represents likelihood to traverse from state to state in Markov Model. APSAi represents the Accomplishment Percentage of Software Artifacts for each artifact. Cost represents the effort t and/or the cost at each certain state. The effort of each state measured in work-hours represents

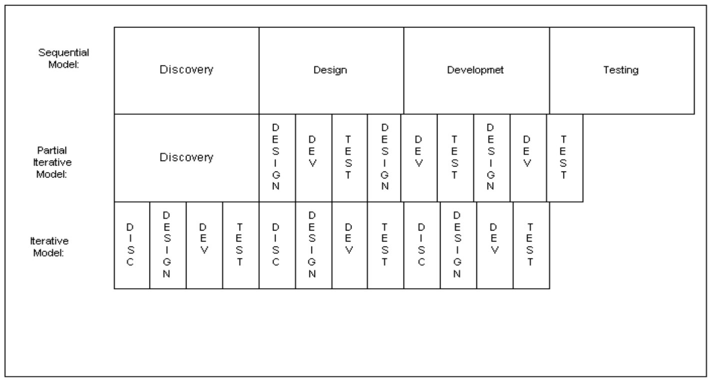


**Figure 1.** Software Engineering Process Model

work expended from the previous state. Software Measurements is widely recognized as an effective means to monitor, control, predict, and improve software development projects. However, effective software measurement requires a great deal of information, data recording be documented.

This paper assumes the framework is based on the notion that software organization has a software measurement environment structured along the following points:

1-Goals and objectives are set relative to the software product and processes.

2- Data collection processes and recording mechanisms are defined and used.

3- A data analysis and corrective action process is defined and used.

4- Measurements and reports are part of closed-loop system that provides current (operational) and historical information to technical staff and management.

These points are prerequisites for all measurement environments, and are stated here to emphasize that their implementation is essential for the successful use of the framework used.

The framework used to satisfy these criteria consists of the following steps:

1-Identify the principal attributes that characterize the object to be measured.

2- Identify the principal classes of values within each attribute.

3- Prepare a checklist of principal attributes and their values.

4- Record the values on a suitable format.

5- Make and record measurements according to the definition and data specifications

One good feature of the model that its ability to use any lifecycle models like sequential model, or spiral model. In addition, any software artifacts can be used and applied in the model. More explanation will be provided later in this paper.

## 5. SOFTWARE ENGINEERING PROCESS MODEL

Process models vary significantly from organization to organization. The software engineering process model may have discovery, phase, design phase, development phase testing phase. A software engineering lifecycle is the process of applying each of these phases, sequentially or iteratively, until a product is no longer supported. [12]. In the sequential paradigm, each phase is performed in order until the product is complete. The process starts in the discovery phase and then moves to each subsequent phase when the prior phase determined to be complete. The iterative model allows the phases to be performed in sequence on a smaller slice of the system. This process is repeated on other parts of the system until the entire system is developed. Figure1 depicts the sequential model and two different versions of the iterative model. In general software process is composed of the following as Whittaker in [13] stated:

1-Phases define the major focus of activity during a particular part of the process.

2-Cycles are sub-phases in which related activity is performed iteratively toward common goal. Thus, several related activities are performed repeatedly until the cycle of work is complete.

3-Tasks define action items and work during a particular phase.

4- Artifacts are the documents and deliverables produced during process execution. Artifacts describe the result of the tasks [13]

## 6. BUILDING THE PROPOSED MODEL

After collecting data of historical projects, the second step is to transfer these data into Markov Chain model from the derived set (M1,M2,…,Mm) depending upon the size and complexity. Based on size and complexity, organization may have one or more models.

The Markov chain is a couple (S, $\delta$) where S is the set of all state of the past software project metrics. These metrics include the APSA, the Accomplishment Percentage of Software Artifacts. $\delta : S \times I\ [0.1] \rightarrow S$ is the non deterministic transition function, where I is the set of function of time f(t). The transition function $\delta$ describes how f(t) cause state changes within the model.

Each state represents APSA for each used artifacts.

The probability distribution associated with each state can represent the Expert Judgment profile, i.e. the probability that the corresponding function of time will be applied. There exists sufficient literature on arriving at judgment parameters in the field of Decision Modeling [12]. The assessment of probabilities describes the likelihood of events. It is useful to quantify the likelihood using probability scale. The expression of probabilities is to quantify the likelihood using probability scale. That would be helpful in allowing project manager to identify decision strategy.

Now, if organization has a new project (Pnew) to be built, a suitable Markov Chain model should be selected from the earlier set (M1,M2,…,Mm). The selection will be based on Analytical Hierarchy Process (AHP).

AHP provides a proven, effective means to deal with complex decision making and can assist with identifying and weighting selection criteria, analyzing the data collected for the criteria and expediting the decision-making process. AHP helps capture both subjective and objective evaluation measures of the new project, providing a useful mechanism for checking the consistency of the evaluation measures and alternatives suggested by the team.

The first step is for the team to decompose the goal into its constituent parts, progressing from the general to the specific. In its simplest form, this structure comprises a goal, criteria and alternative levels. Each set of alternatives would then be further divided into an appropriate level of detail, recognizing that the more criteria included, the less important each individual criterion may become. In the model, the alternatives will be the models that defined by Boehm [14]:

**Table 1.** COCOMO standard size

| Model | Small | Interme diate | Medium | Large | V Large |
|---|---|---|---|---|---|
| Size | 2 KDSI | 8 KDSI | 32 KDSI | 128 KDSI | 512 KDSI |

Next, assign a relative weight to each Model. Each criterion has a local (immediate) and global priority. The sum of all the criteria beneath a given parent criterion in each tier of the model must equal one. Its global priority shows its relative importance within the overall model. The criteria are different from organization to another. An example of criteria will be:

- Size of project
- Business Criticality
- Stability of requirements
- Ease of Communication
- Maturity of Technology
- Performance Constraints
- Reengineering Factors

Next, after the criteria are weighted and the information is collected, put the information into the model. Scoring is on a relative basis, not an absolute basis, comparing one choice to another. Relative scores for each choice are computed within each leaf of the hierarchy. Scores are then synthesized through the model, yielding a composite score for each choice at every tier, as well as an overall score.

The project schedule provides a road map for a software project manager. If it has been properly developed, the project schedule defines the tasks and milestones that must be tracked and controlled as the project proceeds. Tracking can be accomplished in a number of different ways:

- Conducting periodic project status meetings in which each team member reports progress and problems.
- Evaluating the results of all reviews conducted throughout the software engineering process.
- Determining whether formal project milestone have been accomplished by the schedule date.
- Comparing the actual effort and time to planned time and effort for each project.
- Obtain assessment of progress to date.

Cost variance and schedule variance techniques will be used for controlling a project. Both are defined in such a way that they will be negative when the project is behind schedule or over cost. So the effort variances are calculated as he actual effort minus the planned one. The schedule variance is the difference between the planned time and the actual time.

The variances are formulated as ratios rather than differences so that the effort variance become Effort Performance Index (EPI) = Ei / AEi , the Schedule Performance Index (SPI) = Ti/ATi. Use of ratios is particularly helpful when an organization wishes to compare the performance of several projects. However, the accuracy and usefulness of all these measures depend on the degree in which estimates of percent completion reflect reality

## 7. ILLUSTRATIVE EXAMPLE

In this example, all data are mainly collected from organization X that uses formal data collection approach. This organization has maintained and collected a database of all past projects (P1,P2,…,Pn). It uses the iterative model shown in Figure1. The data collection approach should contain information of the product and software artifacts used in each phase of software process. This approach assumes that there is a quantitative technique of recording the history of a product. Therefore, the provided example should have useful techniques for gathering and applying metrics information to cost estimation.

For each project (Pi), the following metrics should be collected:

The data consists of:

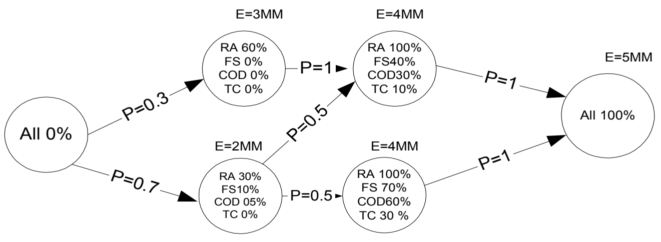
1-Function of time f(t) which is the duration (days) to transfer from state to state. It is fixed for all arcs in each model. However, f(t) varies from model to model.

2- Pi represents the probability to traverse from state to state in Markov Model

3-APSAi represents the accomplishment Percentage of Software Artifacts for each artifact. In this example, RA is the requirement analysis which is an artifact used in discovery phase. FS, functional specification, is design phase artifact. COD, coding, is development phase artifact. TC. Test cases, is testing phase artifacts.

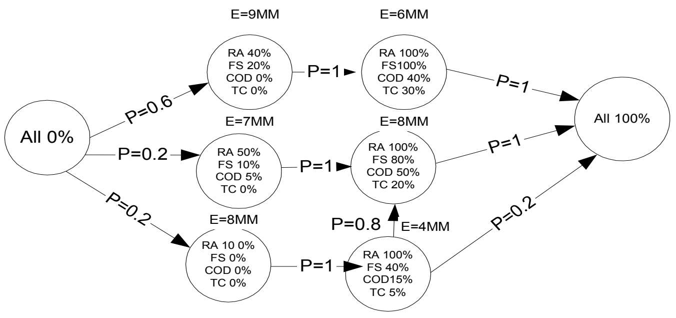
4- Ei represents the effort of each state measured in work-hours represents work expended from the previous state.

**Figure 2.** Model 1 (Small Model)



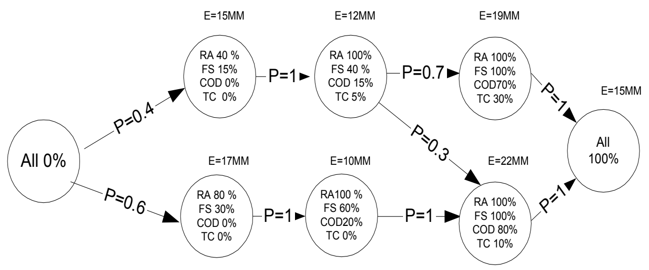
f(t)= 10 days.

**Figure 3.** Model 2 (Medium Model)



f(t)= 25 days

**Figure 4-** Model 3 (Large Model)



f(t)= 60 days

Now, if organization X has a new project (Pnew) to be built, a suitable Markov Chain model should be selected from the set (M1,M2,M2) shown in Figure 2,3, and 4. The selection will be based on Analytical Hierarchy Process (AHP). After applying the AHP, (Pnew will be compared to one of the models. Organization X can compute the following for the new project (Pnew):

- The ability to compute the Effort (Ei) at any certain stage.
- Determine all possible paths Pnew can take.
- Calculate the shortest path with its associated probability.
- Calculate the cost variance and the schedule variance
- Determine whether formal project milestone has been accomplished by the schedule date.

## ACKNOWLEDGMENT

The author extendhis appreciation to the Deanship of Scientific Research at King Saud University for funding this work through the Research Project n**o**NFG2-14-33.

## REFERENCES:

1. *Kemerer, C.F.*, An empirical validation of software cost estimation models, *CACM,***36(2)**, (1993)
2. *Ricardo de A. Araújo, Adriano L.I. Oliveira, Sergio Soares*, A shift-invariant morphological system for software development cost estimation, Expert Systems with Applications, **38 (4)**, 4162-4168, (2011).
3. *NikolaosMittas, Lefteris Angelis*, Visual comparison of software cost estimation models by regression error characteristic analysis, Journal of Systems and Software, **83 (4)**, 621-637, (2010).
4. *Javier Aroba, Juan J. Cuadrado-Gallego, Miguel-Ángel Sicilia, Isabel Ramos, Elena García-Barriocanal*, Segmented software cost estimation models based on fuzzy clustering, Journal of Systems and Software, **81 (11)**, 1944-1950, (2008).
5. *NikolaosMittas, MarinosAthanasiades, Lefteris Angelis*, Improving analogy-based software cost estimation by a resampling method, Information and Software Technology, **50 (3)**, 221-230, (2008).
6. *Ricardo de A. Araújo, Sergio Soares, Adriano L.I. Oliveira*, Hybrid morphological methodology for software development cost estimation, Expert Systems with Applications, **39 (6)**, 6129-6139, (2012).
7. *K. Vinay Kumar, V. Ravi, Mahil Carr, N. Raj Kiran,* Software development cost estimation using wavelet neural networks, Journal of Systems and Software, **81 (11)**, 1853-1867, (2008).
8. *Zhihao Chen, Tim Menzies, Dan Port, and Barry Boehm,* Feature subset selection can improve software cost estimation accuracy, SIGSOFT Softw. Eng. Notes, **30 (4)**, 1-6, (2005).
9. *Qin, X. and M. Fang* Summarization of Software Cost Estimation,Procedia Engineering **15(0)**, 3027-3031, (2011)
10. *ZeynabAbbasiKhalifelu, FarhadSoleimanianGharehchopogh,* Comparison and evaluation of data mining techniques with algorithmic models in software cost estimation, Procedia Technology, **1**, 65-71, (2012).
11. *Ricardo de A. Araújo, Adriano L.I. Oliveira, Sergio Soares, Silvio Meira,* An evolutionary morphological approach for software development cost estimation, Neural Networks, **32**, 285-291, (2012).
12. *Nemecek, S.* ,Systematic defects in software cost-estimation models harm, Management of Engineering and Technology, **1**, 414-415, (2001).
13. Whittaker, James A. Introduction to Software Engineering. Melbourne, Florida: SES Press, 1998.
14. Boehm, B.W., Software Engineering Economics. Prentice-Hall: Englewood Cliffs, NJ, 1981.