International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.4, July-Aug. 2012 pp-2377-2383 ISSN: 2249-6645

On the Estimation of the Software Process Maturity using COCOMO II's Effort Estimation based on CMMI

Yousef Alkouni Alghdr[1], M. Afshar Alam[2], Muqeem Ahmed[3]
Department of Computer Science,Jamia Hamdard Delhi India
Department of computer science Jamia Millia Islamia Delhi India

Abstract: *Software resource estimation methods and models have had a major impact on successful software engineering practice. They provide milestone budgets and schedules that help projects determine when they are making satisfactory progress and when they need corrective action. The software capability maturity model is a most popular model to enhance software processes with the goal of developing best quality of the software which are under the control of budget and schedule. The last stage of updating of the software cost estimation model, constructive cost model has a different set of seventeen cost drivers and a set of five scale factors. The Process maturity is one of the important five scale factors whose ratings are based on the software capability maturity model. This paper is an attempt to determine the effect of process maturity on the software development effort by deriving a new set of constructive cost model II's process maturity rating values based on the most recent version of CMM, i.e., capability maturity model integration. The effect of the constructive cost model II's process maturity scale factor is determined by considering the ideal scale factor methodology. Precedentedness shows all prediction accuracies compared to the generic, constructive cost model II estimation.*

I. Introduction

The Software is one the most important and yet one of the most economically challenging technologies of the current era. As a purely intellectual product, it is among the most labor intensive, complex, and error-prone technologies in human history [1]. The software industry is not untouched by the quality by the quality movement that dramatically affected the product of other industries. But constant demand from the industry for cheaper and better software, make this goal (quality of software) more challenging. Each and every company knows, to remain competitive, it must deliver quality product at time and within budget[2].Consequently many software companies has turned to software process improvement as a way of enhancing the quality of their products ,reducing the cost and accelerating the development process. The delivery of the software on time, within the budget and with the expected functionalities and quality is a challenge for all software development organizations. Inaccurate estimations in software development industry is one of the most serious problems that cause the software failure[2].It is also well known that the quality of software products depends on the software development capabilities and the quality of the maintenance process to a great extent. Thus an organization have no possesses well-trained developers, it will be difficult to them to build the foundation which supports successful improvement of the software development process. Therefore, the fundamental way of ensuring the software quality is to improve the software productivity of the enterprises. And the software productivity of the enterprises depends on their software development capability, especially the maturity of software development and production. Software cost estimation is the process of predicting the effort required to develop a software [3].These types of process becomes one of the major challenges which also the most expensive component in software development. While software cost estimation may be simple in concept, it is difficult and complex in reality [4]. The different estimation models have been developed; most of them have disappeared without any kind of rigorous evaluation. The main reason for that was that these types of models were not good and precise enough [5]. The other reason was that the people who are working in the software development prefer to use their own estimation techniques rather than improving and applying the work of the others. The most of the organizations have relied on experience and ''Price-to-win'' strategies for getting past competitors. Despite the emergence of concepts like because of the rapidly changing technologies, the Software Capability Maturity Model one can never rely completely on experience based estimation in the software industry which renders the experience-based estimates ineffective. The price-to-win strategy is not very favorable for most of the organization [6].Hence the requirement of effective cost model arises to account for the effort spent on the developing software systems. It is an important input to software cost estimation models. The Capability Maturity Model for software was enveloped by Software Engineering Institute to describe the principles and practices underlying software process maturity. Its aim is to help organizations improve their software process maturity through an evolutionary path and process predictability [7]. Despite the fact that the Software Engineering Institute has released the Capability Maturity Model Integration, which is the updated version of the original CMM, COCOMO II still relies on SW-CMM to assess its process maturity scale factor.

This paper is an attempt to describes the effect of process maturity on software development effort by deriving a new set of constructive cost model II's process maturity rating values based on the most recent version of CMMI capability maturity model integration.

II. Review of Literature

The Software development become an important part for many organizations; software estimation is gaining an ever increasing importance in effective software project management. Boehm was the first researcher who considered the software estimation from an economic point

of view, and came up with the cost estimation model, COCOMO in 1981, after investigating a large set of data in the 1970's (Boehm, A, and Chulani,2000). Putnam also developed an early model known as, the Software Lifecycle Management, (Putnam, 1978).The software estimation includes the effort/schedule estimation, quality estimation, risk analysis, etc. The most accurate software estimation can provide powerful assistance for software management decisions (Boehm, 2000). The most new computational techniques are used for cost estimation that are non-algorithmic in the 1990's. The researchers have turned their attention to different approaches which are based on soft computing methods that include artificial neural networks, fuzzy logic models and genetic algorithms. The Artificial neural network is able to generalize from trained data set. Over a known set of training data, a neural-network learning algorithm constructs rules that fit the data and predicts previously unseen data in a reasonable manner (Schofield, 1998). The most popular estimation methods are discussed in detail by Khatibiand J (2011). The COCOMO, SLIM and Albrect's function point methods that measures the amount of functionality in a system were all based on linear regression techniques by collecting data from historical project as the major input to their models. The different algorithmic methods are deliberated as the most popular methods and many researchers used the selected algorithmic methods (Musilek, et al. 2002; Yahiya, et al. 2008; Lavazza and Garavaglia 2009; Yinchuan et al. 2009; Sikka et al. 2010).The Software estimation techniques can support the planning and tracking of software development projects. The efficiently controlling the expensive investment of software development is of prime importance (Gray, MacDonell and Gray, 1997; Jingzhou and Guenther, 2008; Kastro and Bener, 2008; Strike et al., 2001). ImanAttarzadeh and Siew Hock Ow (2010) proposed amodels which is based on COCOMO II and fuzzy logic to the NASA dataset and found that the proposed model performed better than ordinary COCOMO II model and also achieved the results which were closer to the actual effort. The relative error for proposed model using two-side Gaussian membership functions is found to be lower than that of the error obtained using ordinary COCOMO II. A novel neuro-fuzzy Constructive Cost Model is used for software cost estimation and this model carried some of the desirable features of a neuro-fuzzy approach, such as learning ability and good interpretability, while maintaining the merits of the COCOMO model (XishiHuang et al., 2005).

III. Background

A. COCOMO II Model

The COCOMO was originated in 1981[8], and became one of most popular cost estimation models of the 1980s. But the COCOMO faced different difficulties in the 90s, and different complications in cost estimation of software that were developed to a new life cycle processes such as non-sequential and rapid development process models, reuse-driven approaches, and object-oriented approaches [9].Thus, COCOMO II was published initially in the annals of software engineering in 1995 with three sub models; an application-composition model, an early design model and a post-architecture model. The COCOMO II has, as an input, a set of seventeen Effort Multipliers or cost drivers which are used to adjust the nominal effort to reflect the software product being developed. The seventeen COCOMO II factors are shown in Table 1 [10].

a. Effort Estimation

The equation a is formulated the COCOMO II effort estimation model. The effort estimates by both early design and post-architecture models. The inputs are the size of software development, a constant *A*, an exponent *E*, and the number of Effort Multipliers (EM). The number of effort multipliers depends on the model being used.

$$PM = A \times SIZE^{E} \times \prod_{i=1}^{N} EM_i \qquad \text{(a)}$$

Where the constant A=2.94, and the exponent *E* will be described in the bellow.

b. Scale Factors

The study accomplished by [12] presents the conclusion that the most critical input to the COCOMO II model is size, so, a good size estimate is very important for any good model estimation. The Size in COCOMO II is consider as a special cost driver, so it has an exponential factor, E. The exponent *E* in equation 2 is an aggregation of five scale factors. All scale factors have rating levels. These rating levels are Very Low, Low, Nominal, High, Very High and Extra High. Each rating level has a weight W, which is a quantitative value used in the COCOMO II model. The five COCOMO II scale factors are shown in Table1

$$E = B + 0.01 \times \sum_{j=1}^{N} SF_j \qquad \text{(b)}$$

Where *B* is a constant = 0.91. *A* and *B* are constant values devised by the COCOMO team by calibrating to the actual effort values for the 161 projects currently in COCOMO II database.

Table 1. Scale factors of COCOMO II.

Scale Factor	Description
Precedentedness	Reflects the previous experience of the organization.
Development Flexibility	Reflects the degree of flexibility in the development process.
Risk Resolution (RESL)	Reflects the extent of risk analysis carried out.
Team Cohesion	Reflects how well the development team knows each other and work together.
Process Maturity	Reflects the process maturity of the organization.

Table 2. Cost drivers of COCOMO II.

Cost Driver	Description
RELY	Required Software Reliability
DATA	Data base size
RUSE	Developed for Reusability
DOCU	Documentation needs
CPLX	Product Complexity
TIME	Execution Time Constraints
STOR	Main storage Constraints
PVOL	Platform Volatility
ACAP	Analyst Capability
PCAP	Programmer Capability
APEX	Application Experience
PLEX	Platform Experience
LTEX	Language and Tool Experience
PCON	Personnel Continuity
TOOL	Use of Software Tools
SITE	Multisite Development
SCED	Required Development Schedule

The procedure for determining procedure maturity which is the factor of interest in this study- is organized around the SEI-CMM, Table 3

Table 3. The rating levels ,values, and Process Maturity Scale Factor.

Process maturity Description	CMM Level 1 (lower)	CMM Level 1 (upper)	CMM Level 2	CMM Level 3	CMM Level 4	CMM Level 5
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Values	7.80	6.24	4.68	3.12	1.56	0.00

The CMM level 1 is for organizations that don't focus on processes or documenting lessons learned. The CMM level 1is for organizations that have implemented most of the requirements that would satisfy CMM level 2. In CMM's published definition, level 1 (lower half) and (Upper half) are grouped into level 1.

B. The CMM Based Process Maturity

The Software Engineering Institute at Carnegie-Mellon University published the CMM is used to rate an organization's process maturity [11]. It provides a number of requirements that all organizations can use in setting up the software processes used to control software product development. There are five levels of process maturity, level 1 (lowest half) to level 5 (highest). The CMM specifies "what" should be in the software process rather than "when" or "for how long". To be rated at a particular level, the organization should demonstrates capabilities in a set of Key Process Areas associated with a specific CMM level. The capabilities demonstrated in moving from lower levels to higher levels are cumulative. For example, level 3 organizations should show compliance with all key process areas in levels 2 and 3. The CMM process maturity framework is presented in Table 4

Table 4. The Framework of CMM.

CMM Level	Key Process Area
Level 1	None
Level 2 Repeatable	Requirements Management
	Software Project Planning
	Software Project Tracking and Oversight
	Software Subcontract Management
	Software Quality Assurance
	Software Configuration Management
Level 3 Defined	Organization Process Focus
	Organization Process Definition
	Training Program
	Integrated Software Management
	Software Product Engineering
	Intergroup Coordination
	Peer Reviews
Level 4 Managed	Quantitative Process Management
	Software Quality Management
Level 5 Optimizing	Defect Prevention
	Technology Change Management
	Process Change Management

All the organizations are supposed to start at level 1. Which is known as Initial level? At this level, few processes are defined, and the success depends on individual effort which makes the software process unpredictable because it changes as work progresses. Project Schedules, budgets, functionality, and product quality are also unpredictable. Every key process area has a set of goals, capabilities, key practices, measurements and verification practices. The goals state the scope, boundaries, and intent of a key process area. A key practice describes "what" should happen in that key process area. There are a total of 52 goals and 150 key practices.

IV. Methodology

The methodology of our work is the primary data collection tool was a questionnaire that has been used to collect a data from individual projects, i.e., each and every questionnaire should be applied only on one of the project. The questionnaire is based on "COCOMO II cost estimation"

A. The Data Collection

The data collection procedures 55 questionnaires distributed to 20 software development organizations, 35 questionnaires were returned. Some of the questionnaires could not be verified by project managers or senior project

staff; therefore, 16 questionnaires were rejected and eliminated from this study. Therefore, 40 questionnaires were analyzed. The datasets were returned from different fields like banking, insurance, communication, simulation, web development, *etc.* The questionnaires were distributed to software organizations that have already achieved one of the CMMI levels, and spanned the range of its levels, from level 1) to level 4, i.e., 8 data points were collected from each level. For each project, there was a meeting with the project manager or team leader for each project, who would be filling out the forms, in order to clarify each question to ensure that it was well understood and each manager would answer consistently.

B. The Data Analysis

The questionnaires were checked for consistency and went through a data validation process, based on some constraints determined in [6]. There are four aspects that would be extracted and computed: for each questionnaire.

- The set of seventeen COCOMO II's cost drivers. To deal with these seventeen cost drivers, we computed their multiplication. A sample of the cost drivers is shown in Table 5.

- The set of five exponential scale factors. To deal with these five scale factors, we computed their summation. A sample of these scale factors is shown in Table 6 (excluding the last row).

- The Actual effort in Person Months, which extracted for the person hours, as shown in Table 7.

- We collected the project size as a thousand lines of code (KLOC), which is the baseline size in COCOMO II.

We applied equation 1 To predict the effort in person month, which is the basic COCOMO II's formula [6]. At last this analysis, we got the estimated effort for the generic COCOMO II as well as the actual effort for this project.

C. Ideal Scale Factor Analysis on Process Maturity

Boehm[7] has described normalization method out contaminating effects of individual cost driver attributes in order to get clear picture of that cost driver's contribution. Since we have relatively similar situation, i.e., we need to normalize out contaminating effects of a scale factor in process maturity rather than a cost driver. Therefore, for the given project P, compute the estimated development effort using the COCOMO II estimation procedure, with one exception: do not include the value for the Scale Factor Attribute being analyzed. Call this estimate PM (P, Scale Factor Attributes). Then the Ideal Scale Factor, for this project/scale-factor combination is defined as the value which, if used in COCOMO II, would make the estimated development effort for the project equal to its actual development effort PM (P, Actual). i.e.,

Ideal scale factor(P, Process Maturity) =
PM (P, Actual) / PM (P, Process Maturity) (c)

Where Ideal Scale Factor (P, Process Maturity): the ideal scale factor on Process Maturity for project P.
PM (P, Actual): the actual development effort for the project P.
PM (P, Process Maturity): COCOMO II estimate excluding the Process Maturity Scale Factor.

We performed the following steps to complete the Ideal scale factor-Process maturity analysis on our datasets:

1. The first step Compute the *PM* (P, Scale Factor Attribute), using the following formulas:

$$PM = A \times SIZE^{E} \times \Pi EM_i \quad (i=1 \text{ to } 17) \qquad (d)$$

where *A* is a model constant, EM is a set of seventeen effort multipliers as shown in Table 1, and

$$E = B + 0.01 \times \sum_{j=1}^{4} SF\_But\_ProMat_j \qquad (e)$$

Where *B* is a model constant, and scale factor_But the process maturity refers to scale factors except Process Maturity, including Precentedenes REC, Flexibility, Resolution, and Team.

2. Compute the ideal scale factor (P, Scale Factor Attributes) using equation 6.
3. Group Ideal Scale Factor (P, Scale Factor Attributes) by the current CMM process maturity rating (i.e., VL, L, N, H, VH).
4. Compute the mean value for each group as ideal scale factor -Process maturity value for that rating.

This step involves the computation of the mean value of ideal scale factor-process maturity for each CMM rating level.

D. The Prediction Accuracy evaluation

The purpose of this paper is on the degree to which the model's estimated effort measured in Person-Month matches the actual effort. If the model is perfect then for any project, Process maturity =Person month matches. The most common criterion for the evaluation of the cost estimation models is the Relative Error or the Magnitude of Relative Error, which are shown below :

Relative Error= (Person month matches-process maturity)/process maturity (f)

Magnitude of relative Error=(Person month Matches – Process Maturity) $\big|$ / Process Maturity (g)

The Relative Error and Magnitude of Relative Error values are calculated for each project whose effort is predicted. Another criterion that is commonly used is the percentage of predictions that fall within *P* % of the actual, denoted as Predictions *(P)* [13],

Prediction (P) = K / N (h)

K is the number of projects where magnitude of relative error is less than or equal to *P*, and *N* is the number of projects. According to the [10], a standard method for assessing the COCOMO performance is prediction. Therefore we used this criterion to assess the COCOMO II performance as compared to the proposed model. Table 5 through Table 10 shows samples of the calculated data, that represents one project from our forty datasets.

Table 5. COCOMO II Effort Multipliers with their Cost Drivers

Cost Driver	Value
RELY	1.1
DATA	1
RUSE	1
DOCU	1.21
TIME	1.26
STOR	1.01
PVOL	0.77
ACAP	0.61
PCAP	0.68
PCON	0.7
APEX	0.71
PLEX	0.75
LTEX	0.74
TOOL	0.68
SITE	0.76
SCED	1
CPLX	1.21

Table 6. Scale factors of COCOMO II and their values.

Scale Factor	Value
Precedentedness	2.62
Flexibility	1.01
Risk Resolution	1.83
Team Cohesion	1.19
Process Maturity	.59
New process maturity	.03

Table 7. The actual time, effort, size, estimated time, and the cost drivers multiplication.

Description	Value
Actual Time	165
Actual Effort	133.32
Size (KSLOC)	110
Estimated Time, T	173
Π Cost Drivers, EM	0.344

Table 8. The generic COCOMO II Estimated effort.

Description	Value
∑ Scale Factors,	10.210
Estimated Effort,	158.04
Magnitude Relative Error	0.17

Table 9. Ideal Scale Factor and Estimated effort without process maturity value.

Description	Value
∑ Scale Factors-BUT-Process Maturity	9.75
Estimated Effort, but-Process Maturity	156.14
Ideal Scale Factor,	0.92

Table 10. Estimated effort with new process maturity values.

Description	Value
∑ scale factors with ISF-process maturity	10.78
Estimated Effort with ISF-process maturity	163.87
Magnitude Relative **Error=**	0.14

The new set of process maturity rating values under CMMI derived by applying our methodology to the datasets in Table 11.

Table 11. The New Process Maturity Rating values.

Process maturity Description	CMMI Level 1 lower	CMMI Level 1 Upper	CMMI Leve2	CMMI Level 3	CMMI Level 4	CMMI Leve5
Rating level	Very low	low	Nominal	High	Very high	Extra High
New Process Maturity Values	6.44	3.45	2.34	1.34	.98	0.0

In the Figure 1, X axis represents the projects used in CMMI level 4 organization in our study and the Y axis represents the effort. Each project has three columns: the left column represents the actual effort, the middle column represents the generic COCOMO II effort estimation, and the right column represents the effort estimation for the proposed COCOMO II model with new Ideal Scale Factor - Process Maturity Values.

The figure shows that how the proposed model give an estimated effort which is closer to the actual effort than generic COCOMO II estimations due to some data anomalies, especially for low levels companies that do not have good and precise documentations for their historical projects. This case is not absolute, i.e., in some little cases like in CMMI level 1 (lower and upper) and level 2 datasets, the estimated efforts by the generic COCOMO II were relatively closer to the actual effort than the proposed model's estimation.

The proposed model here is uniformly overestimated the effort for most of the 8 projects, so it could still be a consistent model.

The black line in Figure 2 shows the current process maturity scale factor values used in COCOMO II. It shows that an increase in process maturity level corresponds with a reduction in project effort. It shows the new process maturity values derived from the ideal scale factor-process maturity analysis.

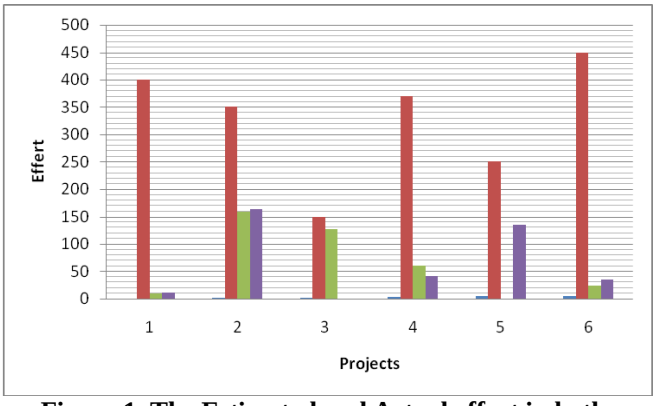

Figure 1. The Estimated and Actual effort in both generic COCOMO II and COCOMO II with Ideal Scale Factor-Process Maturity.

The few number of different Process Areas are assigned to this level, and success still depends on individual effort that is very low and low rating levels in COCOMO II's procedure maturity are categorized under CMMI level 1, Therefore, level 1 companies still need much effort to accomplish their projects, particularly for CMMI level 1 companies that rely on "heroes" to do the jobs and do not show any compliance that would satisfy subsequent levels.

The second observation is the nominal and high rating levels demonstrated a relatively obvious reduction in the procedure maturity values, which appears as a deviation first line in Figure 2. The underlying explanation behind this reduction might be due to the major additions and refinements that have occurred at CMMI maturity levels 2 and 3. As an example, going from seven key process areas in CMM level 3, to 14 process areas in CMMI level 3, and just two were dropped. These additions and refinements in maturity levels 2 and 3 reflect their significance and definitely will reduce the effort required to develop the software systems in CMMI maturity levels 2 and 3 organizations.

Ideal Scale Factor Process Maturity VS COCOMO II Process Maturity

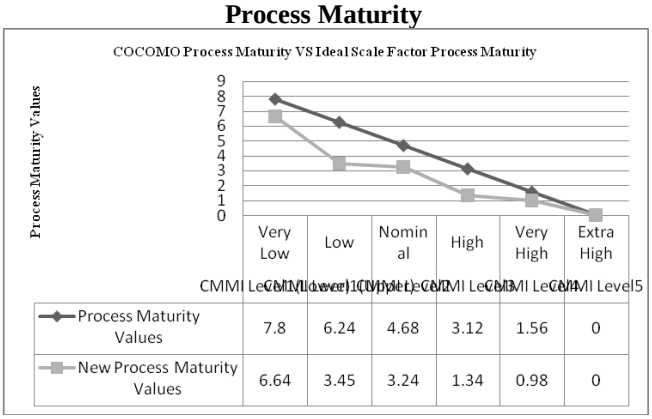

Figure 2. The Ideal scale factor -process maturity values. vs COCOMO II's process maturity values.

E. The Results of the Model Accuracy with ideal scale factor

The improvement in the model's accuracy has been realized which is shown in the below after applying the derived ideal scale factor-process maturity values back to our datasets, in the below Table 12.

Table 12. The Analysis of Accuracy Results

Level	Generic COCOMO II	COCOMO II with New Process Maturity Values	Improvement
Level 1 (Lower)	43%	55%	9%
Level 1 (Upper)	30%	43%	11%
Level 2	18%	55%	34%
Level 3	18%	58%	47%
Level 4	30%	77%	21%

The above Table shows the analysis of accuracy results that by applying the ideal scale factor procedure maturity values into our datasets which had been collected from CMMI organizations, the accuracy in all maturity levels increased by 9%, 11%, 34%, 47%, and 21% respectively as we have already mentioned and justified, that level 3 has the highest percentage of improvement, and the level 1has lowest percentage of improvement.

V. Conclusions

The software development cost estimation is very important in all the aspects of project such as budgeting, planning and effective control of management. There are various software cost estimation models which have various inputs. The most important inputs to software cost estimation models is the process maturity. According to survey, the present values for the COCOMO II process maturity scale factor does not adequately reflect the impact of CMMI-based process maturity on development efforts. Therefore, by using the ideal scale factor method and with the aid of our datasets, we have identified the new process maturity values which better reflect the impact of CMMI based process maturity on software development effort. The new values provide an improvement in COCOMO II model accuracies 9% for CMMI level one, 11% for CMMI level one, 34% for CMMI level two, 47% for CMMI level three, and 21% for CMMI level four organizations. In future the amount of datasets allocated to each CMMI maturity level could be expanded to get a clearer picture of the impact of CMMI-based process maturity on software development effort and the locally calibrating the proposed model parameters to a particular organization, which requires collecting data from more projects belonging to the same Organization.

References

[1] Miyazaki Y. and Mori K., "COCOMO Evaluation and Tailoring," *in Proceedings of ICSE 8, IEEE-ACM-BCS*, pp. 292-299, 1985.

[2] Al-Sakran H., "Software Cost Estimation Model Based on Integration of Multi Agent and Case Based Reasoning," *Journal of Computer Science*, vol. 2, no. 3

[3] OLGA F, LEONOR T, HELENA A." Software Effort Estimation with Multiple Linear Regression: review and practical application" Journal of Information Science and Engineering xx, xxx-xxx 2011

[4] Jones C. "Software Cost Estimation in 2002,"Computer Journal of Defense Software Engineering, vol. 15, no. 6, pp. 4-8, 2002.

[5] Miyazaki Y. and Mori K. "COCOMO Evaluation and Tailoring," Proceedings of ICSE 8, IEEE-ACM-BCS, pp. 292-299, 1985.

[6] Dillibabu R. and Krishnaiah K., "Cost Estimation of a Software Product Using COCOMO II.2000 Model a Case Study," International Journal of Project Management, vol. 23, no. 2, pp. 297-307, 2005.

[7] Jianguo Li, Jinghui Li, and Hongbo Li," Research on Software Process Improvement Model Based on CMM" World Academy of Science, Engineering and Technology 39 2008

[8] Boehm B., "Software Engineering Economics", Prentice Hall, 1981.

[9] Boehm B., Clark B., Horowitz E., Westland C., Madachy R., and Selby R., "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," Proceedings of Special Volume on Software Process and Product Measurement, Amsterdam, pp. 45-60, 1995.

[10] Boehm B., Horowitz E., Madachy R., Reifer D., Clark B., Steece B., Brown A., Chulani S., and Abts C.,"Software Cost Estimation with COCOMO II", Prentice Hall, 2000.

[11] Yang Y. and Clark B., "COCOMO II.2003 Calibration Status," http://sunset. usc.edu /events /2003/ March_ 2003 /COCOMO_ II_2003_ Recalibration.

[12] Musilek P., Pedrycz W., Sun N., and Succi G., "On the Sensitivity of COCOMO II Software Cost Estimation Model," Proceedings of the 8th IEEE Symposium on Software Metrics, IEEE Computer Society, Washington, pp. 13, 2002.

[13] Conte S., Dunsmore H., and Shen V., "Software Engineering Metrics and Models, "Menlo Park, CA, Benjamin/Cummings, 1986.