

Software Cost Estimation Methods: A Review

¹Vahid Khatibi, ²Dayang N. A. Jawawi

^{1,2}Faculty of Computer Science and Information System Universiti Teknologi Malaysia (UTM), Johor, Malaysia

¹khatibi78@yahoo.com, ²dayang@utm.my

ABSTRACT

Project planning is one of the most important activities in software projects. Poor planning often leads to project faults and dramatic outcomes for the project team. If cost and effort are determined pessimistic in software projects, suitable occasions can be missed; whereas optimistic predictions can be caused to some resource losing. Nowadays software project managers should be aware of the increasing of project failures. The main reason for this problem is imprecision of the estimation. In this paper, several existing methods for software cost estimation are illustrated and their aspects will be discussed. Comparing the features of the methods could be applied for clustering based on abilities; it is also useful for selecting the special method for each project. The history of using each estimation model leads to have a good choice for an especial software project. In this paper an example of estimation is also presented in an actual software project.

Keywords- *Cost Estimation; project fail; Cocomo; Accuracy*

I. INTRODUCTION

Several indicators should be considered to estimate the software cost and effort. One of the most important indicators which should be noticed is the size of the project. The estimation of effort and cost depends on the accurate prediction of the size. Generally, the effort and cost estimations are difficult in the software projects. The reason is that software projects are often not unique and there is no background or previous experience about them. Therefore, prediction seems complicated. On the other hand, production in such projects is not tangible so the measurement of effort, cost and the amount of progress in the software project is very difficult. In addition, requirements of the software projects are changing continuously which will cause changing of the prediction. Because of the mentioned problems, project managers usually try to avoid from using cost or effort estimations or at least do the estimations at a limited domain.

Describing the usage and importance of the estimation methods and their effect on the project success seems necessary in the software projects so that the software project managers are ensured of their usage. Inaccuracy in the software cost and effort estimation via optimistic or pessimistic prediction may cause many problems in the software projects. The main objective of this paper is demonstrating the abilities of the software cost estimation methods and clustering them based on their features which makes helps readers to better understanding. The full paper is organized as follows: in section Three, after Introduction and Background, we briefly overview the estimation techniques. Section four includes the comparison of existing methods. An actual example is presented in section five and finally, the conclusion is illustrated in section six.

II. BACKGROUND

Software project failures have been an important subject in the last decade. Software projects usually don't fail during the implementation and most project fails are related to the planning and estimation steps. Despite going to over time and cost, approximately between 30% and 40% of the software projects are completed and the others fail (Molokken and Jorgenson, 2003). The Standish group's CHAOS reports failure rate of 70% for the software projects (Glass, 2006). Also the cost overrun has been indicated 189% in 1994 CHAOS report (Jørgensen and Molokken-Ostfold, 2006). Glass (2006) claims the reported results do not depict the real failures rate and are pessimistic. In addition, Jørgensen and Molokken-Ostfold, (2006) indicate that the CHAOS report may be corrupted. Nevertheless the mentioned statistics show the deep crisis related to the future of the software projects. (Glass, 2006; Jørgensen and Molokken-Ostfold, 2006). During the last decade several studies have been done in term of finding the reason of the software projects failure. Galorath and Evans (2006) performed an intensive search between 2100 internet sites and found 5000 reasons for the software project failures. Among the found reasons, insufficient requirements engineering, poor planning the project, suddenly decisions at the early stages of the project and inaccurate estimations were the most important reasons. The other researches regarding the reason of project fails show that inaccurate estimation is the root factor of fail in the most software project fails (Jones, 2007; Jørgensen, 2005; Kemerer, 1987; Molokken and Jørgensen, 2003). Despite the indicated statistics may be pessimistic, inaccurate estimation is a real problem in the software production's world which should be solved. Presenting the efficient techniques and reliable models seems required regarding the mentioned problem. The conditions of the software projects are not stable and the state is continuously changing so several methods should be presented for

estimation that each method is appropriate for a special project.

III. ESTIMATION TECHNIQUES

Generally, there are many methods for software cost estimation, which are divided into two groups: Algorithmic and Non-algorithmic. Using of the both groups is required for performing the accurate estimation. If the requirements are known better, their performance will be better. In this section, some popular estimation methods are discussed.

A. Algorithmic Models

These models work based on the especial algorithm. They usually need data at first and make results by using the mathematical relations. Nowadays, many software estimation methods use these models. Algorithmic Models are classified into some different models. Each algorithmic model uses an equation to do the estimation:

$$\text{Effort} = f(x_1, x_2, \dots, x_n) \quad (1)$$

where, $(x_1 \dots x_n)$ is the vector of the cost factors. The Differences among the existing algorithmic methods are related to choosing the cost factors and function. All cost factors using in these models are:

- Product factors: required reliability; product complexity; database size used; required reusability; documentation match to life-cycle needs;
- Computer factors: execution time constraint; main storage constraint; computer turnaround constraints; platform volatility;
- Personnel factors: analyst capability; application experience; programming capability; platform experience; language and tool experience; personnel continuity;
- Project factors: multisite development; use of software tool; required development schedule.

Quantizing the mentioned factors is very difficult to do and some of them are ignored in some software projects. In this study several algorithmic methods are considered as the most popular methods. The mentioned methods have been selected based on their reputation. There are many papers which use the selected algorithmic methods (Musilek, Pedrycz et al. 2002; Yahya, Ahmad et al. 2008; Lavazza and Garavaglia 2009; Yinhuan, Beizhan et al. 2009; Sikka, Kaur et al. 2010)

1) Source Line of Code

SLOC is an estimation parameter that illustrates the number of all commands and data definition but it does not include instructions such as comments, blanks, and continuation lines. This parameter is usually used as an analogy based on an approach for the estimation. After

computing the SLOC for software, its amount is compared with other projects which their SLOC has been computed before, and the size of project is estimated. SLOC measures the size of project easily. After completing the project, all estimations are compared with the actual ones.

Thousand Lines of Code (KSLOC) are used for estimation in large scale. Using this metric is common in many estimation methods. SLOC Measuring seems very difficult at the early stages of the project because of the lack of information about requirements.

Since SLOC is computed based on language instructions, comparing the size of software which use different languages is too hard. Anyway, SLOC is the base of the estimation models in many complicated software estimation methods. SLOC usually is computed by considering S_L as the lowest, S_H as the highest and S_M as the most probable size (Roger S. Pressman, 2005).

$$S = \frac{S_L + 4S_M + S_H}{6}$$

2) Function Point Size Estimates

At first, Albrecht (1983) presented Function Point metric to measure the functionality of project. In this method, estimation is done by determination of below indicators:

- User Inputs,
- User Outputs,
- Logic files,
- Inquiries,
- Interfaces

A Complexity Degree which is between 1 and 3 is defined for each indicator. 1, 2 and 3 stand for simple, medium and complex degree respectively. Also, it is necessary to define a weight for each indicator which can be between 3 and 15.

At first, the number of each mentioned indicator should be tallied and then complexity degree and weight are multiplied by each other. Generally, the unadjusted function point count is defined as below:

$$UFC = \sum_{i=1}^5 \sum_{j=1}^3 N_{ij} W_{ij} \quad (3)$$

where N_{ij} is the number of indicator i with complexity j and; W_{ij} is the weight of indicator i with complexity j . According to the previous experiences, function point could be useful for software estimations because it could be computed based on requirement specification in the early stages of project. To compute the FP, UFC should be multiplied by a Technical Complexity Factor (TCF) which is obtained from the components in Table I.

TABLE I Technical Complexity Factor components

F1	Reliable back-up and recovery	F8	Data communications
F2	Distributed functions	F9	Performance
F3	Heavily used configuration	F10	Online data entry
F4	Operational ease	F11	Online update
F5	Complex interface	F12	Complex processing
F6	Reusability	F13	Installation ease
F7	Multiple sites	F14	Facilitate change

Each component can change from 0 to 5. 0 and 5 indicate that the component has no effect on the project and the component is compulsory and very important respectively. Finally, the TCF is calculated as:

$$\text{TCF} = 0.65 + 0.01(\text{SUM}(\text{Fi})) \quad (4)$$

The range of TCF is between 0.65 (if all F_i are 0) and 1.35 (if all F_i are 5). Ultimately, Function Point is computed as:

$$\text{FP} = \text{UFC} * \text{TCF} \quad (5)$$

3) Seer-Sem

SEER-SEM model has been proposed in 1980 by Galorath Inc (Galorath, 2006). Most parameters in this method are commercial and, business projects usually use SEER-SEM as their main estimation method. Size of the software is the most important feature in this method and a parameter namely S_e is defined as effective size. S_e is computed by determining the five indicators : newsize, existingsize, reimpl and retst as below:

$$S_e = \text{Newsize} + \text{ExistingSize}(0.4\text{Redesign} + 0.25\text{reimp} + 0.35\text{Retest}) \quad (6)$$

After computing the S_e the estimated effort is calculated as below:

$$\text{Effort} = t_d = D^{-0.2} \times \left(\frac{S_e}{C_{te}} \right)^{0.4} \quad (7)$$

where D is relevant to the staffing aspects; it is determined based on the complexity degree in staffs structure. C_{te} is computed according to productivity and efficiency of the project method is used widely in commercial projects. (Fischman,L.This ,2005)

4) Linear Models

Commonly linear models have the simple structure and trace a clear equation as below:

$$\text{Effort} = a_0 + \sum_{i=1}^n a_i x_i \quad (8)$$

where, a_1, a_2, \dots, a_n are selected according to the project information.

5) Multiplicative models

The form of this model is as below:

$$\text{Effort} = a_0 \prod_{i=1}^n a_i^{x_i} \quad (9)$$

where, a_1, \dots, a_n are selected according to the project information . In this formula, only allowed values for x_i are -1, 0 and +1.

6) COCOMO

Cost models generally use some cost indicators for estimation and notice to all specifications of artifacts and activities. COCOMO 81 (Constructive Cost Model), proposed by Barry Boehm (Boehm, 1981), is the most popular method which is categorized in algorithmic methods. This method uses some equations and parameters, which have been derived from previous experiences about software projects for estimation. COCOMO-II is the latest version of COCOMO that predicts the amount of effort based on Person-Month (PM) in the software projects.

It uses function point or line of code as the size metrics and composes of 17 Effort Multipliers (shown in Table II) and 5 scale factors (shown in Table III). Some rating levels are defined for scale factors including very low, low, nominal, high, very high and extra high. A quantitative value is assigned to each rating level as its weight.

COCOMO II has some special features, which distinguish it from other ones. The Usage of this method is very wide and its results usually are accurate.

Two equations are used to estimate effort and schedule as below:

$$\begin{aligned} \bullet PM_{NS} &= A * \text{Size}^E * \prod_{i=1}^{17} EM_i \\ \text{Where } E &= B + 0.01 * \sum_{j=1}^5 SF_j, A = 2.94, B = 0.91 \end{aligned} \quad (10)$$

$$\begin{aligned} \bullet TDEV &= C * (PM)^F \\ \text{Where } F &= D + 0.2 * 0.01 * \sum_{j=1}^5 SF_j = D + 0.2 * (E - B) \\ C &= 3.67, D = 0.28 \end{aligned} \quad (11)$$

TABLE II Effort Multipliers

Attribute	Type	Description
RELY	Product	Required system reliability
CPLX	Product	Complexity of system modules
DOCU	Product	Extent of documentation required
DATA	Product	Size of database used
RUSE	Product	Required percentage of reusable components
TIME	Computer	Execution time constraint
PVOL	Computer	Volatility of development platform
STOR	Computer	Memory constraints
ACAP	Personnel	Capability of project analysts
PCON	Personnel	Personnel continuity
PCAP	Personnel	Programmer capability
PEXP	Personnel	Programmer experience in project domain
AEXP	Personnel	Analyst experience in project domain
LTEX	Personnel	Language and tool experience
TOOL	Project	Use of software tools
SCED	Project	Development schedule compression
SITE	Project	Extent of multisite working and quality of inter-site communications

TABLE III Scale factors

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

7) Putman's model

This model has been proposed by Putman according to manpower distribution and the examination of many software projects (Kemerer,2008). The main equation for Putnam's model is:

$$S = E \times (\text{Effort})^{1/3} t_d^{4/3} \quad (12)$$

where, E is the environment indicator and demonstrates the environment ability. T_d is the time of delivery. Effort and S are expressed by person-year and line of code respectively. Putnam presented another formula for Effort as follows:

$$\text{Effort} = D_0 \times t_d^3 \quad (13)$$

where, D_0 , the manpower build-up factor, varies from 8(new software) to 27(rebuilt software). By combining equations 12 and 13, the final equation is obtained as:

$$\text{Effort} = (D_0^{4/7} \times E^{-9/7}) \times S^{9/7} \quad (14)$$

$$t_d = (D_0^{-1/7} \times E^{-3/7}) \times S^{3/7} \quad (15)$$

SLIM (Software Life Cycle Management) is a tool that acts according to the Putnam's model.

B. Non Algorithmic Methods

Contrary to the Algorithmic methods, methods of this group are based on analytical comparisons and inferences. For using the Non Algorithmic methods some information about the previous projects which are similar the under estimate project is required and usually estimation process in these methods is done according to the analysis of the previous datasets. Here, three methods have been selected for the assessing because these methods are more popular than the other None Algorithmic methods and many papers about their usage have been published in the recent years(Idri, Mbarki et al. 2004; Braz and Vergilio 2006; Li, Xie et al. 2007; Keung, Kitchenham et al. 2008; Li, Lin et al. 2008; Jianfeng, Shixian et al. 2009; Jorgensen, Boehm et al. 2009; Pahariya, Ravi et al. 2009; Cuadrado-Gallego, Rodri et al. 2010).

1) Analogy

In this method, several similar completed software projects are noticed and estimation of effort and cost are done according to their actual cost and effort. Estimation based on analogy is accomplished at the total system levels and subsystem levels. By assessing the results of previous actual projects, we can estimate the cost and effort of a similar project. The steps of this method are considered as:

- i. Choosing of analogy
- ii. Investigating similarities and differences
- iii. Examining of analogy quality
- iv. Providing the estimation

In this method a similarity function is defined which compares features of two projects. There are two popular similarity function namely Euclidean similarity (ES) and Manhattan similarity (MS) (Shepperd, M, Schofield, C,1997, Chiu , N, Huang, S.J,2007).

$$Sim(p, p') = 1 / \left[\sqrt{\sum_{i=1}^n w_i Dis(f_i, f'_i)} + \delta \right] \quad \delta = 0.0001 \quad (16)$$

$$Dis(f_i, f'_i) = \begin{cases} (f_i - f'_i)^2, & \text{if } f_i \text{ and } f'_i \text{ are numeric or ordinal} \\ 1 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\ 0 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i \neq f'_i \end{cases}$$

p and p' are projects w_i is the weight is assigned to each feature and varies between 0 and 1. F_i and F'_i display the ith feature of each project and n demonstrates the number of features. δ is used for obtaining the none zero results. The MS formula is very similar to ES but it computes the absolute difference between features.

$$Sim(p, p') = 1 / \left[\sum_{i=1}^n w_i Dis(f_i, f'_i) + \delta \right] \quad \delta = 0.0001 \quad (17)$$

$$Dis(f_i, f'_i) = \begin{cases} |f_i - f'_i| & \text{if } f_i \text{ and } f'_i \text{ are numeric or ordinal} \\ 1 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\ 0 & \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i \neq f'_i \end{cases}$$

2) Expert judgment

Estimation based on Expert judgment is done by getting advices from experts who have extensive experiences in similar projects. This method is usually used when there is limitation in finding data and gathering requirements. Consultation is the basic issue in this method. One of the most common methods which works according to this technique, is Delphi. Delphi arranges an especial meeting among the project experts and tries to achieve the true information about the project from their debates. Delphi includes some steps:

- i. The coordinator gives an estimation form to each expert.
- ii. Each expert presents his own estimation (without discussing with others)
- iii. The coordinator gathers all forms and sums up them (including mean or median) on a form and ask experts to start another iteration.
- iv. steps (ii-iii) are repeated until an approval is gained.

Figure 1 shows an example of using Delphi technique in which eight experts contributed and final convergence was determined after passing four stages (Mahmud S et al. ,2008).

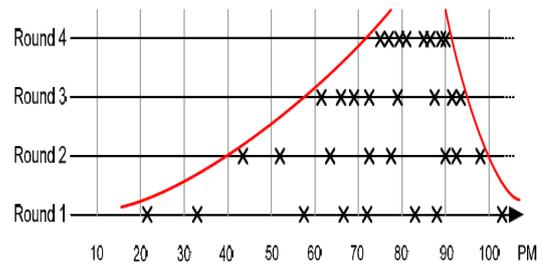


Figure1. An example of using Delphi

3) Machine learning Models

Most techniques about software cost estimation use statistical methods, which are not able to present reason and strong results. Machine learning approaches could be appropriate at this filed because they can increase the accuracy of estimation by training rules of estimation and repeating the run cycles. Machine learning methods could be categorized into two main methods, which are explained in the next subsections.

a) Neural networks

Neural networks include several layers which each layer is composed of several elements called neuron. Neurons, by investigating the weights defined for inputs, produce the outputs. Outputs will be the actual effort, which is the main goal of estimation.

Back propagation neural network is the best selection for software estimation problem because it adjusts the weights by comparing the network outputs and actual results. In addition, training is done effectively. Majority of researches on using the neural networks for software cost estimation, are focused on modeling the Cocomo method, for example in (Attarzadeh, Ow, 2010) a neural network has been proposed for estimation of software cost according to the following figure. Figure 2 shows the layers, inputs and the transfer function of the mentioned neural network. Scale Factors(SF) and effort multipliers(EM) are input of the neural network, p_i and q_j are respectively the weight of SFs and EMs.

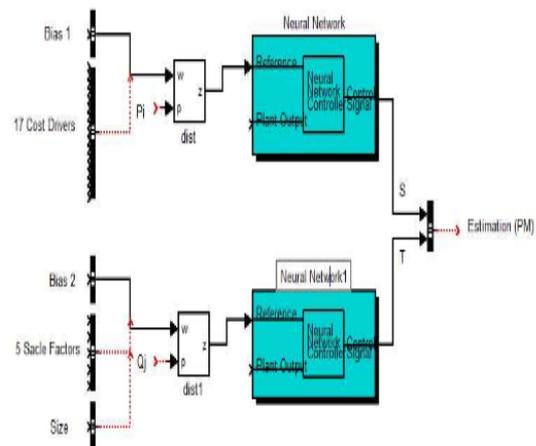


Figure2. An example of using Neural Network

b) Fuzzy Method

All systems, which work based on the fuzzy logic try to simulate human behavior and reasoning. In many problems, which decision making is very difficult and conditions are vague, fuzzy systems are an efficient tool in such situations. This technique always supports the facts that may be ignored. There are four stages in the fuzzy approach:

Stage 1: Fuzzification: to produce trapezoidal numbers for the linguistic terms.

Stage 2: to develop the complexity matrix by producing a new linguistic term.

Stage 3: to determine the productivity rate and the attempt for the new linguistic terms.

Stage 4: Defuzzification: to determine the effort required to complete a task and to compare the existing method.

For example in (Attarzadeh, Ow, 2010) Cocomo technique has been implemented by using fuzzy method. The Figure 3 displays all mentioned steps; In the first step fuzzification has been done by scale factors, cost drivers and size . In step 2, principals of Cocomo are considered and defuzzification is accomplished to gain the effort.

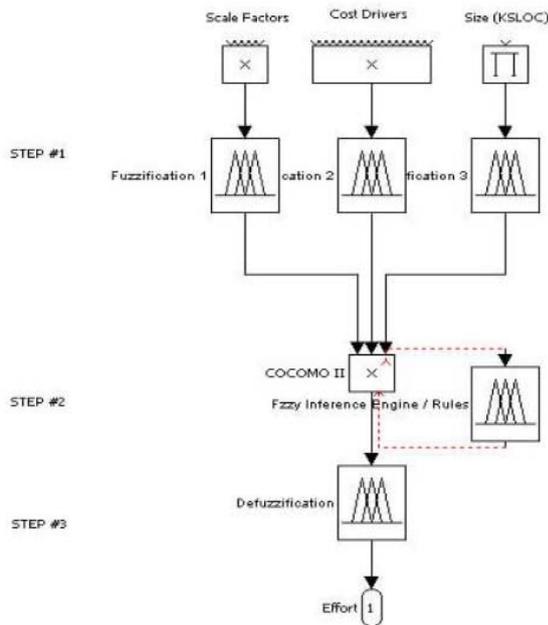


Figure 3. An example of using Fuzzy method

IV. COMPARISON OF THE ESTIMATION METHODS

At this section according to the previous presented subjects, it is possible to compare mentioned estimation methods based on advantages and disadvantages of them. This comparison could be useful for choosing an appropriate method in a particular project. On the other hand, selecting the estimation technique is

done based on capabilities of methods and state of the project. Table IV shows a comparison of mentioned methods for estimation. For doing comparison, the popular existing estimation methods have been selected.

TABLE IV Comparison of the existing methods

Method	Type	Advantages	Disadvantages
COCOMO	Algorithmic	Clear results, very common	Much data is required, It 's not suitable for any project,
Expert Judgment	Non-Algorithmic	Fast prediction, Adapt to especial projects	Its success depend on expert, Usually is done incomplete
Function Point	Algorithmic	Language free, Its results are better than SLOC	Mechanization is hard to do , quality of output are not considered
Analogy	Non-Algorithmic	Works based on actual experiences, having especial expert is not important	A lots of information about past projects is required, In some situations there are no similar project
Neural Networks	Non-Algorithmic	Consistent with unlike databases, Power of reasoning	There is no guideline for designing, The performance depends on large training data
Fuzzy	Non-Algorithmic	Training is not required, Flexibility	Hard to use, Maintaining the degree of meaningfulness is difficult

According to the current comparison and based on the principals of the algorithmic and non algorithmic methods(described in previous sections); for using the non algorithmic methods it is necessary to have the enough information about the same previous projects because these methods perform the estimation by analysis of the historical data. Also, non algorithmic methods are easy to learn because all of them follow the human behavior. On the other hand, Algorithmic methods are based on mathematics and some experimental equations. They are

usually hard to learn and they need to the much data about the current project state. But if enough data is reachable, these methods present the reliable results. In addition, algorithmic methods usually are complementary to each other, for example, Cocomo uses the SLOC and Function Point as two input metrics and generally if these two metrics are accurate, the Cocomo presents the accurate results too. Finally, for selecting the best method to estimate, looking at available information of the current project and the same previous projects data could be useful.

V. HOW TO EVALUATE THE RESULTS OF AN ESTIMATION METHOD

After knowing estimation methods and comparing them with each other, illustrating their abilities via some actual projects seems to be useful. The acceptance of using these methods has been a challenge since many years ago. In this section, we try to show the minimum distance between estimated parameters and actual ones in an experience. Measuring the performance of estimation methods is accomplished by computing several metrics including RE (Relative Error), MRE (Magnitude of Relative Error) and MMRE (Mean Magnitude of Relative Error). They are the popular parameters which are used for performance evaluation of methods.

$$RE = (\text{Estimated PM} - \text{Actual PM}) / (\text{Actual PM}) \quad (18)$$

$$MRE = |\text{Estimated PM} - \text{Actual PM}| / (\text{Actual PM}) \quad (19)$$

$$MMRE = \sum MRE \quad (20)$$

Another parameter used for performance evaluation is PRED (Percentage of the Prediction) which is calculated as:

$$PRED(X) = A/N \quad (21)$$

where, A is the number of projects with MRE less than or equal to X and N is the number of considered projects.

A. An actual estimation by Cocomo

Since CCOCOMO II is the most popular method used for estimation, in this section, a real project effort estimation is demonstrated based on COCOMO II metrics. Table V shows the cost drivers and their adjusted amounts which are related to a real project. In (Maged A. Yahya et al., 2008) all data about thirty software projects has been gathered from organizations by distributing the questionnaire. The scope of activities in mentioned organizations is banking, insurance, web development, communication and so on. For illustrating executive capabilities of Cocomo, we used only one project data of thirty projects which has been recorded in the dataset.

TABLE II Effort Multipliers

Cost Driver	Degree	Value
RELY	Very High	1.1
DATA	Very high	1.28
RUSE	Very High	1.15
DOCU	Nominal	1
TIME	High	1.11
STOR	High	1.05
PVOL	Low	0.87
ACAP	Very high	0.71
PCAP	Very High	0.76
PCON	Very high	0.81
APEX	Very high	0.81
PLEX	Very High	0.85
LTEX	Very High	0.84
TOOL	Very High	0.78
SITE	Very High	0.86
SCED	Nominal	1
CPLX	High	1.17

Before using Cocomo II formulas, it is necessary to adjust scale factors. Table VI shows the scale factors which are arranged for this project.

TABLE VI. Scale Factors

Scale Factor	Level	Value
PREC	High	2.48
FLEX	Nominal	3.04
RESEL	High	2.83
TEAM	Very High	1.10
PMAT	Very High	1.56

Real effort for this project is 117.92 person-month, real time is 15.5 months and size of the project (KSLOC) is 130. Now by using cost drivers, scale factors and relations 10, 11 and 19 parameters are estimated as:

$$PM = 2.94 * (130)^{0.91+0.01*11.01} * 0.326 = 137.4$$

$$TDEV = 3.67 * (137.4)^{0.28+0.2*0.01*11.01} = 16.23$$

$$MRE = (137.4 - 117.92) / 117.92 = 0.17$$

As it is seen, the difference of actual parameters and estimated ones is very low somehow the MRE is 0.17 which is the reasonable value. It was only a sample to

show the applicable ability of COCOMO method. By calibration, the accuracy of metric results in this method will be increased.

VI. CONCLUSION

Finding the most important reason for the software project failures has been the subject of many researches in last decade. According to the results of several researches presented in this paper, the root cause for software project failures is inaccurate estimation in early stages of the project. So introducing and focusing on the estimation methods seems necessary for achieving to the accurate and reliable estimations. In current study most of the present estimation techniques have been illustrated systematically. Since software project managers are used to select the best estimation method based on the conditions and status of the project, describing and comprising of estimation techniques can be useful for decreasing of the project failures. There is no estimation method which can be present the best estimates in all various situations and each technique can be suitable in the special project. It is necessary understanding the principals of each estimation method to choose the best. Because performance of each estimation method depends on several parameters such as complexity of the project, duration of the project, expertise of the staff, development method and so on. Some evaluation metrics and an actual estimation example have been presented in this paper just for describing the performance of an estimation method (for example Cocomo). Trying to improve the performance of the existing methods and introducing the new methods for estimation based on today's software project requirements can be the future works in this area.

REFERENCES

- [1] Attarzadeh, I. Siew Hock Ow, "Proposing a New Software Cost Estimation Model Based on Artificial Neural Networks", IEEE International Conference on Computer Engineering and Technology (IC CET), Volume: 3, Page(s): V3-487 - V3-491 2010.
- [2] Attarzadeh, I. Siew Hock Ow, "Improving the accuracy of software cost estimation model based on a new fuzzy logic model", World Applied science journal 8(2):117-184, 2010-10-2.
- [3] Albrecht, A.J. and J. E. Gaffney, "Software function, source lines of codes, and development effort prediction: a software science validation", IEEE Trans Software Eng. SE, pp.639-648, 1983.
- [4] Braz, M. R. and S. R. Vergilio. "Software Effort Estimation Based on Use Cases". Computer Software and Applications Conference, COMPSAC '06. 30th Annual International, 2006.
- [5] Boehm, B. W., & Valerdi, R. "Achievements and challenges in cocomo-based software resource estimation", IEEE Software, 25(5), 74-83. doi:10.1109/MS, 2008.
- [6] Boehm, "Software Engineering Economics", Prentice Hall, 1981.
- [7] Chiu, N.H., Huang, S.J., "The adjusted analogy-based software effort estimation based on similarity distances", Journal of Systems and Software 80 (4), 628-640. 2007.
- [8] Cuadrado-Gallego, J. J., Rodri, et al. "Analogies and Differences between Machine Learning and Expert Based Software Project Effort Estimation". Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 11th ACIS International Conference on, 2010.
- [9] Fischman, L. K. McRitchie, and D.D. Galorath, "Inside SEER-SEM", CrossTalk, The Journal of Defense Software Engineering, 2005.
- [10] Galorath, D. D., & Evans, M. W. "Software sizing, estimation, and risk management: When performance is measured performance improves". Boca Raton, FL: Auerbach, 2006.
- [11] Idri, A., S. Mbarki, et al. "Validating and understanding software cost estimation models based on neural networks". Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on, 2004.
- [12] Jones, C. "Estimating software costs: Bringing realism to estimating (2nd ed.)". New York, NY: McGraw-Hill, 2007.
- [13] Jianfeng, W., L. Shixian, et al. "Improve Analogy-Based Software Effort Estimation Using Principal Components Analysis and Correlation Weighting". Software Engineering Conference, 2009. APSEC '09. Asia-Pacific, 2009.
- [14] Jorgensen, M., B. Boehm, et al. "Software Development Effort Estimation: Formal Models or Expert Judgment?" Software, IEEE 26(2): 14-19, 2009.
- [15] Jørgensen, M. "Practical guidelines for expert-judgment-based software effort estimation", IEEE Software, 22(3), 57-63. doi:10.1109/MS. 73, 2005.
- [16] Keung, J. W., B. A. Kitchenham, et al. "Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation." Software

- Engineering, IEEE Transactions on 34(4): 471-484,2008.
- [17] Kemerer, C. "An empirical validation of software cost estimation models", Communications of the ACM, 30(5), 416-429. doi: 10.1145/22899. 22906 ,1987.
- [18] Lavazza, L. and C. Garavaglia "Using function points to measure and estimate real-time and embedded software: Experiences and guidelines". Empirical Software Engineering and Measurement ESEM 2009. 3rd International Symposium on,2009.
- [19] Li, J., J. Lin, et al." Development of the Decision Support System for Software Project Cost Estimation. Information Science and Engineering", 2008. ISISE '08. International Symposium on,2008.
- [20] Li, Y. F., M. Xie, et al. "A study of genetic algorithm for project selection for analogy based software cost estimation. Industrial Engineering and Engineering Management", IEEE International Conference on,2007
- [21] Mahmud S. Alkoffash, Mohammed J. Bawaneh and Adnan I. Al Rabea,"Which Software Cost Estimation Model to Choose in a Particular Project", Journal of Computer Science 4 (7): 606-612, 2008.
- [22] Maged A. Yahya, Rodina Ahmad, Sai Peck Lee," Effects of Software Process Maturity on COCOMO II's Effort Estimation from CMMI Perspective",978-1-4244-2379-8/08 IEEE (c), 2008 .
- [23] Musilek, P., W. Pedrycz, et al. "On the sensitivity of COCOMO II software cost estimation model". Software Metrics, Proceedings. Eighth IEEE Symposium on,2002.
- [24] Moløkken, K., & Jørgensen, M." A review of surveys on software effort estimation. International Symposium on Empirical Software Engineering", 223-231. Retrieved from ACM Digital Library database,2003.
- [25] Shepperd, M., Schofield, C," Estimating software project effort using analogies".IEEE Transactions on Software Engineering 23 (12), 733-743.1997.
- [26] Pressman, Roger S. " Software Engineering: A Practitioner's Approach", 6th Edn., McGraw-Hill New York, USA., ISBN: 13: 9780073019338,2005
- [27] Pahariya, J. S., V. Ravi, et al."Software cost estimation using computational intelligence techniques". Nature & Biologically Inspired Computing, NaBIC 2009. World Congress on,2009.
- [28] Sikka, G., A. Kaur, et al. " Estimating function points: Using machine learning and regression models". Education Technology and Computer (ICETC), 2nd International Conference on,2010.
- [29] Yahya, M. A., R. Ahmad, et al." Effects of software process maturity on COCOMO II's effort estimation from CMMI perspective". Research, Innovation and Vision for the Future, RIVF. IEEE International Conference on,2008.
- [30] Yinhuang, Z., W. Beizhan, et al. "Estimation of software projects effort based on function point". Computer Science & Education. ICCSE. 4th International Conference on,2009.