Volume 4, Issue 8, August 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey of Various cost & Effort Estimation Models

[1]Kuldeep Singh, [2]Professor Upendra Dwivedi
Department of Computer Science and Engineering
Shri Vaishnav Institute of Technology and Science indore (MP), India

Abstract - Software effort and cost estimation is a very crucial activity. If the estimation is done precisely, it results in decreasing error rate. Estimation process represents the reality of project's progress. It avoids cost/budget or schedule overruns. This paper proposes a survey of the most popular cost estimation models. It will propose the pros and cons of every model. It will help future researchers up to a great deal and they will be able to develop more efficient model for the accurate cost estimation. This process is quite simple which takes a few inputs. This assessment framework helps inexperienced team improve project tracking and estimation. Much work can be carried on it. Various COCOMO parameters can be adjusted. Further work can be carried on learning based methods which apply weights to calculation of each software module based on priorities and criticalities. A good estimate should have amount of granularity so it can be explained. Since the effort invested in a project is one of the most important and most analyzed variables.

Keywords— Function Point (FP), Total effort multiplier (TEM), Scale Factors, Cost Drivers.

I. INTRODUCTION

Software cost estimation plays an important role in software engineering practice, often determining the success or failure of contract negotiation and project execution. Cost estimation's deliverables, such as effort, schedule, and staff requirements are valuable pieces of information for project formation and execution

Software maintenance is an important activity in software engineering. Over the decades, software maintenance costs have been continually reported to account for a large majority of software costs. This fact is not surprising. On the one hand, software environments and requirements are constantly changing, which lead to new software system upgrades to keep pace with the changes. On the other hand, the economic benefits of software reuse have encouraged the software industry to reuse and enhance the existing systems rather than to build new one. Thus, it is crucial for project managers to estimate and manage the software maintenance costs effectively.

The success of a software project is determined by various factors that are related each other in the project. A project will be called a success if all the requirements can be fulfilled, the cost is not excessive (overflow), did not pass through the schedule (deadline) that has been planned. In this paper, we will discuss the use of function to measuring the cost estimated of a software system, namely the academic system an educational institution which will be compared on the estimation of the model using the Unified Modeling Language (UML) based object model with the Data Flow Diagram (DFD) based structured model.

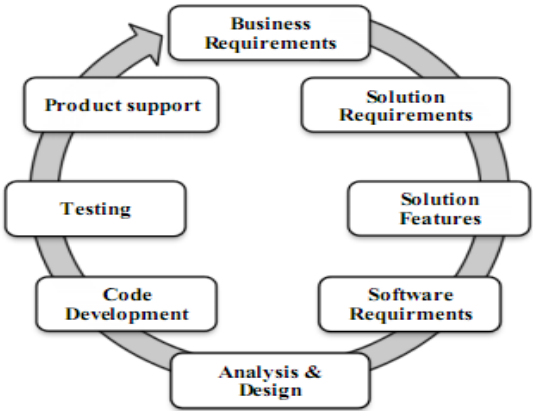

Figure 1: Software Development Life Cycle

Software engineering cost (and schedule) models and estimation techniques are used for a number of purposes . These include:

- Budgeting
- Tradeoff and risk analysis
- Project planning and control
- Software improvement investment analysis

II. LITERATURE SURVEY

Wide range of maintainability prediction models have been proposed in the literature within last two decades. Some of the models are predicting maintainability using the metrics from coding as well as design phase, while some are focusing only on design level metrics [3]. Anton Ellis et al. [4], proposed a method of mapping object oriented source code metrics onto the sub-characteristics of maintainability mentioned in ISO 9126. Oman and Hagemeister [5], proposed the Maintainability Index (MI) that objectively determines the maintainability of software system based upon the status of the source code. Welker and Oman [6], suggested measuring maintainability in terms of cyclomatic complexity, lines of code(LOC) and lines of comments. Hayes et al. [7], proposed a model that estimates Adaptive software maintenance effort in terms of difference lines of code (DLOC) i.e. number of added, deleted and updated lines. Polo et al. [8], used number of modification requests, mean effort per modification request and type of correction to examine maintainability. In another study Hayes and Zhao [9], proposed a maintainability model that categorizes software modules as „easy to maintain" and „not easy to maintain". The model helps the developers to identify the modules those are not easy to maintain, before integrating them. From the survey of literature it has been observed that various researchers proposed several models for maintainability estimation, but in most of these studies, maintainability estimation depends on the measures taken after the coding phase. Because of this, maintainability predictions are made in the latter stages of SDLC, and it became very difficult to improve the maintainability at that stage.

In this paper, we will discuss the use of Adapted module Reused module and new module method to measuring the cost estimated of a software system, namely the academic system an educational institution which will be compared on the estimation of the model using the Unified Modeling Language (UML) based object model with the Data Flow Diagram (DFD) based structured model. Each cost driver has been rated on a six-point ordinal scale ranging from low to high importance. Based on the rating, an effort multiplier is determined, Product of all effort multipliers leads to EAF. Cost drives have a rating level; these rating can range from Extra Low to Extra High. For the purpose of quantitative analysis We are also calculating effort estimation in this paper so we will calculate in this types :

We are maintain effort estimation Using Information domain and the weighting factor Count Total is calculated. The complexity weights are applied to the initial function point count to arrive at an unadjusted point total.

Efort Value adjustment factor is based on the responses to the following 14 general system characteristics.

The fourth group is the quality factor, which is the set off quality characteristics, they are Functionality, Reliability, Usability, Efficiency Maintainability and Portability.

In software Development Intermediate COCOMO model computes very good effort as a function of program size and a set of cost drivers. The cost drivers are assigned new ratings in such a way that the existing characteristic behaviour of the intermediate model is not altered.

Total Effort multiplier is the product of the ratings of the assigned cost drivers. From the obtained TEM, the developmental person month is calculated, which is very much nearer to the planned effort. In case of software maintenance the Intermediate COCOMO model not computes very good effort as a function of program size and a set of cost drivers. So we will used of some modules we modified the software system and our project is good compute cost and effort in case of software maintenance.

Project Manager must know the cost, effort, schedule and functionality in advance. Project factors change in the duration of a project. Significant Research was carried out by Boehm in software cost modelling which began with the extensive 1965 study of the 105 attributes of 169 software project. This led to some useful partial models in the late 1960s and early 1970s. Although much work was carried on developing models of cost estimation, all of them were in same dilemma: as software grew in size and importance it also grew in complexity "It was very difficult to predict the accurate cost of software maintenance." The fast changing nature of software maintenance has made it very difficult to develop parametric models that yield high accuracy for software maintenance in all domains. Software maintenance costs continue to increase and practitioners continually express their concerns over their ability to predict accurately the costs involved.

Muthanna et al. [10], developed a maintainability model using polynomial linear regressions. But this model could be applied only for procedural software and not suitable for object-oriented software. Genero et al. [11], developed four models that relate size and structural complexity metrics of UML class diagrams with maintainability measures like understandability time, modifiability correctness and modifiability completeness. But none of the four models quantify the maintainability of class diagrams itself. Earlier MEMOOD model was developed which finds the maintainability of the class diagrams on the basis of Understandability and Modifiability on the basis of object oriented metrics of class diagrams [12].

III. PROBLEM DEFINITION

COCOMO model is the most common model used for size and effort estimation. Also in industry, it is the most common model for size and effort estimation. Historical data and assumptions of software development projects were used for building the COCOMO model. The properties of COCOMO model such as forms, cost factors, and constants are supposed to be applicable for estimating the cost of software maintenance. However, the software and software maintenance have so many differences. For an example the maintenance of a software generally depends on the architecture of the system, design, source code, and supporting documentation. So there is a problem in using the COCOMO model for software maintenance. Because of the difference between software development and software maintenance the parameters of COCOMO model becomes less relevant for maintenance. Even if we use the COCOMO model for software estimation, then the results are less accurate.

IV. SOLUTION DOMAIN

We will propose a unique method for measuring the size of software reuse & maintenance. This method allows the maintainer to measure the size of both software reuse and maintenance work using the same parameters and formulas. This unique method gives actual size of complete reuse and maintenance work based on source code delivered. It will result in improving software estimation accuracy. In our proposed model, we will consider all three aspects of maintenance: adapted code, new code, and reused code while computing the size of the software. On the other side the COCOMO II model, just takes the KLOC into consideration. Hence our proposed model will give more accurate values of effort and schedule estimates. Because these two depends up on the value of size. So the size is required to be estimated in accurate manner.

COCOMO model gives less accurate size and effort estimates for the maintenance. Thus, my thesis is " Enhanced Size estimates for better Software Maintenance". We will propose efficient estimates.

Proposed Model:

Software maintenance depends on preexisting code. Our proposed model classified the code into three categories:

1. Reused modules :
   Reused module is used to software maintenance in reused oldest software in our thesis.
2. Adapted modules
   Adapted modules is used to software maintenance in in adapt or modify oldest software in our thesis.
3. New modules
   New module is used make a new software in software maintenance in our thesis.

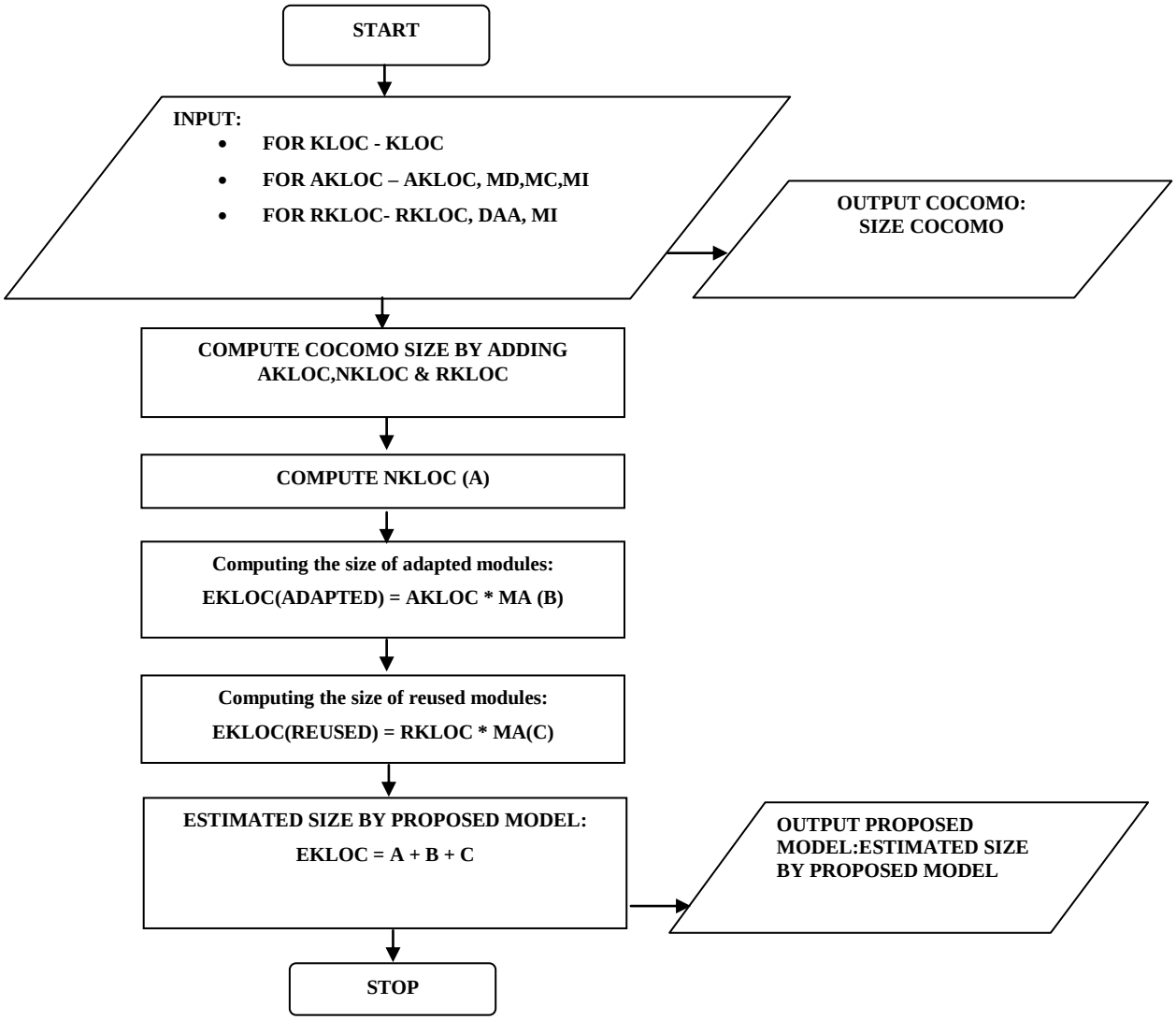

Figure shown the in software maintenance the COCOMO model used only KLOC in the system so COCOMO model gives less accuracy and less relevant result in cost and effort estimation. Other side our proposed model used all three aspect of maintenance: new code, adapted code, and reused code so computing the cost and size all three code and estimated cost and size this unique method gives actual cost and size and good accuracy and relevant. It will improving software estimation accuracy.

Where, NKLOC = New kilo Line of code, AKLOC = Adapted kilo Line of code, RKLOC = Reused kilo line of code, EKLOC = Estimated kilo Line of code MD= model development , MC = matrix complexity , MI = maintainability Index

V. EXPECTED OUTCOMES

This method allows the maintainer to measure the cost and size of both software reuse and maintenance work using the same parameters and formulas. This unique method gives actual cost and size of complete reuse and maintenance work based on source code delivered. And also We will propose a novel method for accurate size estimation, a novel method for effort size estimation, a novel method for accurate Cost estimation , and our unique method is more relevant in software maintenance. It will result in improving software estimation accuracy.so the prediction of this value while we start the software projects, it helps to plan any forthcoming activities adequately. Estimating the effort with large value of realibility is a problem which has not been solved

VI. CONCLUSION

In this paper, we have proposed a survey of modern effort and size estimation models. We have discussed the pros and cons of the present estimation models. We have also proposed a new model. In near future we will come up with the complete model. The unique difference between the proposed and existing cost estimation of effort for the software system development is the level of quality consideration If the estimation is done accurately, it results in error decrease. Estimation process reflects the reality of project's progress. It avoids cost/budget or schedule overruns. That is, existing estimations are using only few quality factors for effort estimation, but the proposed effort estimation covers the ISO 9126 quality factors, since the effort invested in a project is one of the most important and most analyzed variables.

ACKNOWLEDGEMENT

This research paper is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings. Especially, please allow me to dedicate my acknowledgment of gratitude toward the following significant advisors and contributors :

First and foremost, I would like to thank Mr Upendra Dwivedi (Department of CSE in SVITS Indore) for his most support and encouragement. He kindly read my paper and offered invaluable detailed advices on grammar, organization, and the theme of the paper.

Finally, I sincerely thank to my parents, family, and friends, who provide the advice and financial support. The product of this research paper would not be possible without all of them.

REFRENCES

[1] V.S. Alagar, L.Qiaoyum, and O.S. Ormandjieva, "Assessment of Maintainability in Object-Oriented Software," Proc. 39th International Conference and Exhibition on Technology of Object-Oriented Languages andSystems (TOOLS39), 29 July -03 Aug. 2010, vol. 19, pp. 194 – 204, Santa Barbara, Callifornia, USA, 2010.

[2] M. Dagpinar and J. Jahnke, "Predicting Maintainability with Object-Oriented Metrics – an Empirical Comparison," Proc10th Working Conference on Reverse Engineering (WCRE'03),13 - 17 Nov. 2011, pp. 155-164, 2011.

[3] S.W.A. Rizvi and R.A. Khan, "A Critical Review on Software Maintainability Models," Proc. of the National Conference on Cutting Edge Computer and Electronics Technologies, 14 - 16 Feb. 2009, pp. 144 – 148,Pantnagar, India, 2009.

[4] P. Antonellis, D. Antoniou, Y. Kanellopoulos, C. Makris, E. Theodoridis, C. Tjortjis, and N. Tsirakis, "A Data Mining Methodology for Evaluating Maintainability According toISO/IEC-9126 Software Engineering Product Quality.Standard," Proc. 11th IEEE Conference onSoftware Maintenance and Reengineering (CSMR2007), 21 – 23 Mar.2007, Amsterdam, Netherlands, 2007.

[5] P.W. Oman and J.R. Hagemeister, "Construction and Testing of Polynomials Predicting Software Maintainability," Journal of Systems and Software, vol. 24, no. 3, pp. 251- 266, 2013.

[6] K.D. Welker and P.W. Oman, "Software Maintainability Metrics Models in Practice," Journal of Defense Software Engineering, vol. 8, no. 11, pp. 19 - 23, 1995.

[7] J.H. Hayes, S.C. Patel, and L. Zhao, "A Metrics-Based Software Maintenance Effort Model," Proc. 8th European Conference on Software Maintenanceand Reengineering (CSMR'04), 24 – 26 Mar. 2004, pp. 254 – 258,IEEE Computer Society, 2004.

[8] M. Polo, M. Piattini, and F. Ruiz, "Using Code Metrics to Predict Maintenance of Legacy Programs: a Case Study," Proc. of InternationalConference on Software Maintenance, ICSM 2001, pp. 202-208, IEEE Computer Society, Florence Italy, 2001.

[9] J.H. Hayes and L Zhao, "Maintainability Prediction: a Regression Analysis of Measures of Evolving Systems," Proc. 21st IEEE InternationalConference on Software Maintenance, 26 - 29 Sept. 2005, pp. 601 -604, 2005.

[10] S. Muthanna, K. Kontogiannis, K. Ponnambalam, and B. Stacey, "A Maintainability Model for Industrial Software Systems Using Design Level Metrics," Proc. 7th Working Conference on Reverse Engineering(WCRE'00), 23 - 25 Nov., 2000, pp. 248 – 256, Brisbane, Australia, 2000.

11] M. Genero, E. Manso, and G. Cantone, "Building UML Class Diagram Maintainability Prediction Models Based on Early Metrics," Proc. 9th International Symposium on Software Metrics (METRICS'03), 3 - 5 Sept., 2012, pp. 263 - 275, 2012.

[12] S.W.A. Rizvi and R.A. Khan "Maintainability Estimation Model for Object- Oriented Software in Design Phase (MEMOOD)".

