iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** V **Month of publication:** May 2018

DOI: http://doi.org/10.22214/ijraset.2018.5171

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue V, May 2018- Available at www.ijraset.com

Comprehensive Analysis of Effort Estimation in Software Development

Mandeep kaur[1], Navpreet Rupal[2]
[1]*Research Scholar,* [2]*Professor Department of computer Science and Engineering, G.I.M.E.T, Amritsar,*

Abstract: ***Accurate effort estimation of software is the state of the art problem. Setting apart, organization of software project effort, cost and time are fundamental of any software advance way. Checking for the software development in terms of size is an issue, since models such as Cost Constructive Model (COCOMO) are already present but still failure of software known as crisis is vigorous. Software effort estimation if can be estimated in advanced then cost can be predicted and success or failure of software project can be determined in advance. This paper proposed a mechanism by altering the environmental variables within COCOMO model for easy and early identification of cost and effort associated with software project. Time of development is a key issue which, each and every model failed to identify. This paper proposes an additional variable consideration including electricity failure(E_F), Machine failure(M_F) and weather condition(W_C). Considering these parameters, size of organic, semidetached and embedded projects are determined. Schedule although is increased by said mechanism but accuracy is also enhanced.***

Keywords: Cost constructive model, Environmental Factors, Size, Organic, Embedded, Semi detached.

I. INTRODUCTION

Effort estimation is required to yield accurate cost associated with the software system. The software crisis becomes critical analysis under circumstances in which failure in software is often[1].Several models including cost constructive model(COCOMO) was created for checking the overall cost associated with the system[2]. The Constructive Cost model (COCOMO) this is a type of programming software model that is used to decide the cost associated with in the system in advance. In other words, software cost can be determined before software is being created hence helping to determine the feasibility of overall operation[3].It works by joining a regression formula with predefined parameters that are explained through the information of a specific system. The fundamental cost constructive model preferred standpoint is that you can decide the costs that will be associated with in particular system. Another cost constructive model favorable is that the evaluations and all other related data that is obtained is real, so your outcomes are constantly exact. It can also determine the structure of the software by the use of constructive cost model and understand the system[4],[5]. The best cost constructive model advantage is that it can be repeated any number of times, this means that you can calculate the cost of a particular project initially and determine how changes and modifications will affect your initial project estimates. It contains three sub-models, every one offering extended consistency the advance along one is in the arranging sequence and setup process. The proposed system also test various Metric by the use of constructive cost model on algorithmically complex programs to show the error in estimating effort for such program which are getting to be regular today, especially in embedded frameworks programming and Artificial Intelligence based programs. The software cost estimation techniques[6]. COCOMO (Constructive. Cost Model) is the mostly used algorithmic cost modeling technique because of its simplicity for estimating the effort in system at different stages. COCOMO also uses the mathematical formulas to predict cost associated with in the system. Many researchers have searched the possibility of using neural networks for estimating the effort. The most commonly used architecture for estimating software effort is feed forward multilayer perceptron with back propagation learning algorithm[7].Viability of the model is give in terms of the metrics. There are two metrics commonly used in order to estimate the effort required to create a actual system.

A. LOC(Line of Code)

It is a direct measure in which number of lines within the software are counted to determine size and effort required by the software. LOC however cannot be determine in advance hence it cannot be used for future predictions.

B. Function Point metric

Function point metric is a indirect measure depending upon the constant and variable parameters to determine actual cost and effort required to create a software.

This paper provides the accurate effort and cost estimation mechanism within the software system by considering environmental variable which is not considered in existing COCOMO II model. Next section provides the literature review of existing models and techniques used to determine effort required to create software.

II. LITERATURE SURVEY

Software effort estimation becomes critical to minimize software crisis. 90% of the software failure resulting due to timeline violation within which software must be delivered to the user. To estimate effort required to construct software accurately, modeling is used. These models are described as under. N. Gupta et.al suggested that this model is of primary importance since it will check viability of the software product well in advance hence cost and time required to create unacceptable software is saved. Earlier this model is termed as COCOMO 81. There are variations associated with COCOMO model[8].

M. Madheswaran et.al. suggested that in this model software development effort is given as function of software size. In other words cost is directly proportional to size of the software. Program size is estimated in terms of Source LOC and KDLOC[9].

Jun Liu, Zheng Xu, Jianzhong Qiao et.al. suggested by organic projects in which size of the software being created is small hence software team size required is also small. Organic project are comparatively smaller in size and estimating the cost is comparatively easy. Also size leads to effort required which is less in this case. Organic projects thus suffer less from flaws. Organic projects does not have any back log or delayed component associated with it hence modular approach or complex problems are hardly a part of this system.

Semi detached considerably larger team and project size than organic projects is being used in semi detached system. It is comparatively large as compared to organic system. Complex problems may be a part of thus system. Failure as compared to organic projects is common in this case.

To tackle the issue of failures, fault tolerant strategies are employed within the semi detached system. This makes it more expensive as compared to organic system.

Embedded larger team size and effort required is a part of embedded projects. Embedded projects are complex and have higher probability of failures. The embedded projects suffer from enhanced degree of team effort and hence required coordination facility within the users communicating within this group.

Embedded project suffer from software crisis and required to be prevented.

The proposed literature takes into consideration this project types and tries to reduce failure ratio associated with this project types. The estimation equation is derived from. Basic COCOMO model however has limited functionality and is not extendable to changing physical environment. To tackle the issues Intermediate COCOMO Model comes into existence[10].

N. Rutar and J. K. Hollingsworth et.al.

Suggested that intermediate COCOMO model originated from basic COCOMO model with effort required is given in terms of software size and set of cost drivers. The cost drivers used within Intermediate COCOMO model is derived from the online source Wikipedia.

Most of the parameter values are available online but environmental factors are missing which are elaborated within the proposed system these attributes are associated with product, hardware, personal and project attributes [11].

G. Mathew, T. Menzies, and J. Hihn, D. Maksimovic this model is an advancement of intermediate COCOMO model. the COCOMO II model is also known as Phase sensitive model.

The impact of COCOMO along with cost drivers is considered on each individual phase of software development process. The parametric evaluation is detailed in this case. Hence it is more expensive as compared to other COCOMO Models[13],[14].

Jun liu, N. rutar et.al.

suggested that Detailed COCOMO model is again the pre-matured model with lack of environmental variables. The proposed system solves the software crisis problem by accommodating the environmental variables. The effort is estimated accurately since environmental variable gives the advance effort which can be considered later on in the project development.

in all the COCOMO models environmental factors such as weather conditions, machine failures and electricity failure is not consider hence actual effort required my suffer than actual effort[11],[12].

III. PROPOSED WORK

A. Methodology

Algorithm for improved Cost Constructive Model with environmental consideration.

Setup objective function: Effort_Estimation=X|E(where X is the effort estimation variables and E is the environment variables)

1) Initialize the variables and input the type of project

2) Repeat the following steps until $|x - x_i| < 0.001$(where 0.001 is prescribed tolerance)

2.a) model followed for organic project

Effort=$c_1 * (KDSI)^{p1}$

Schedule=C2 * (Effort)^{p2}

Where c1 is the scaling coefficient for effort and c2 is scaling coefficient for schedule

2.b) Semi Detached

Effort in man months = $C1_i$ *EAf * (KDSI)^{p1}

Schedule in total month = C2 * (EFFORT)^{p2}

EAF = E1 * E2* … E15

2.c) Embedded

Intermediate +assessed per phase (analysis, design, etc)

2.d) x_i=EAF

End of loop

3) Stop

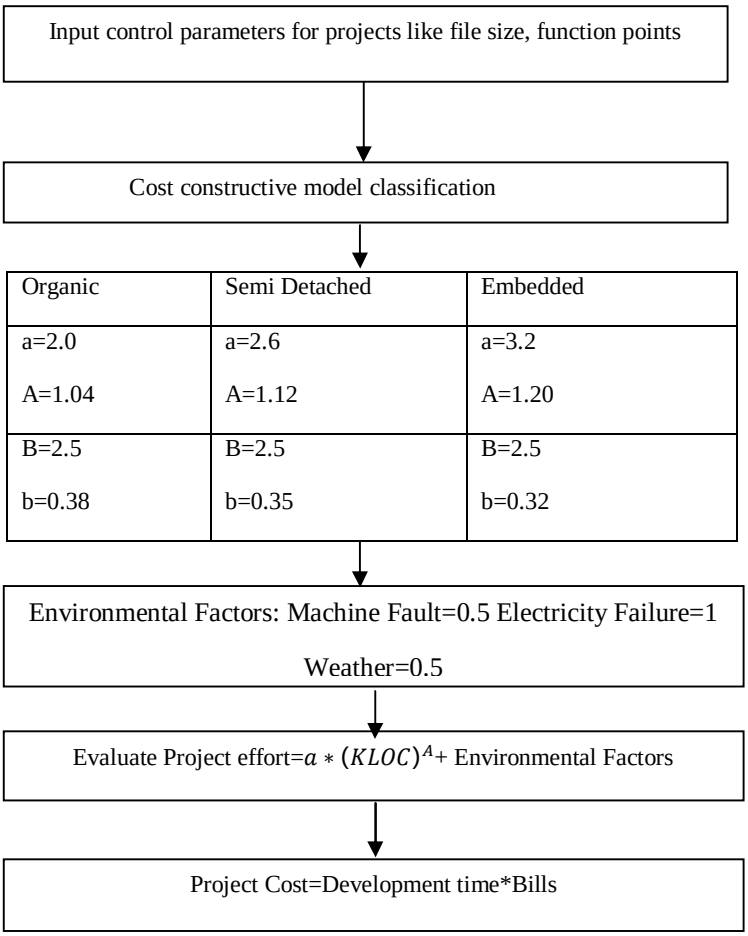


Fig. 1 Functional flow Diagram of proposed work

E. Kocaguneli, T. Menzieset.al. suggested that the proposed system considers environmental factors in addition to the cost drivers to determine the actual cost required to construct a software system. Set of cost drivers are considered for evaluation under proposed system includes product attributes, hardware attributes, personal attributes and project attributes.

These attributes along with environmental condition are mixed or hybridized to obtain the actual cost required in terms of size of the software. The key parameters consider includes[15].

TABLE I. KEY PARAMETERS OF COST FACTORS AND DRIVERS

Cost factors and drivers	Weightage					
	V-L	L	N	H	V-H	E-H
Product Attributes						
Software reliability Requirements	0.7	0.8	1.01	1.15	1.24	--
Database size	--	0.9	1.00	1.07	1.16	--
Product complexity	0.7	0.88	1.01	1.14	1.30	1.66
Hardware properties						
Run time performance conditions	--	--	1.00	1.11	1.30	1.77
Memory constraints	--	--	1.01	1.06	1.21	1.56
Holding capacity of virtual machines	--	0.88	1.00	1.15	1.30	---
Required turnaround time	--	0.87	1.00	1.87	1.00	--
Personal attributes requirement						
Capability to analysis a given product	1.45	1.19	1.00	0.86	0.87	--
Application experience	1.29	1.13	1.00	0.91	0.88	--
Engineer capability	1.42	1.17	1.01	0.86	0.70	-
Virtual machine capability	1.21	1.10	1.00	0.90	--	---
Programming language compatibility	1.14	1.07	1.02	0.95	--	--
Attributes associated with project						
Application of S/w Engg methods	1.24	1.10	1.00	0.91	0.82	--
Use of tools	1.25	1.11	1.00	0.91	0.83	--
Schedule	1.23	1.08	1.00	1.04	1.10	---

These are the parameters and weigtage given within the software project effort estimation mechanism. in addition to these cost drivers, environment factors are also considered in the proposed system. These parameters along with weightage is described as.

TABLE II. ENVIRONMENT FACTORS

Parameters	Weightage
Electricity Failure (E_F)	1
Machine_Failure(M_F)	0.5
Weather Conditions(W_C)	0.5

Highest factor and value is given to electricity failure since electricity failure could rendered entire system unusable. entire system unusable. Values of a and b are 2.4 and 1.05 derived from Wikipedia.

TABLE III. VALUES OF a AND b IN SOFTWARE PROJECT

Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Effort estimation equation which is used in this case is modified as

$$Effort = a(KLOC)^b * Enviornment_{factor} \text{.....(i)}$$

Where Environment Factor is given as

Environment factor=E_C+M_F+W_C ……(ii)

Effort estimation through the proposed system is highly accurate effort estimation that could lead to success or failure of system at early stage of the software.

B. Performance Analysis

The performance is analyzed in terms of effort required effort required depends greatly on the size of the software being used.

TABLE IV. PERFORMANCE ANALYZED IN TERM OF EFFORT

Project Type	Effort Required(Person per month)
Organic	1.45
Semi detached	2.34
Embedded	3.67

The plot for the project and effort required is given as under

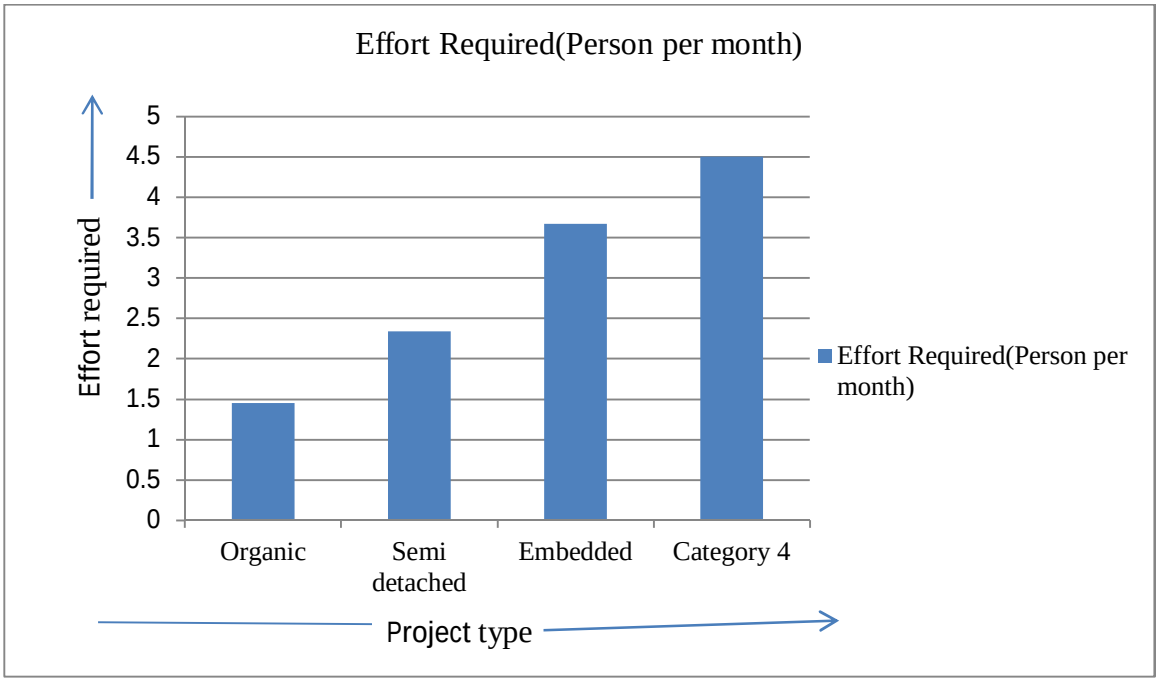


Fig.2 Effort Required in project

The performance analysis indicates that by considering the environmental factors time consumption is increased. This time required indicates in case of any anomaly effort required also enhances. In other words effort required is directly prepositional to the anomalies along with size of the software project.

Comparison with respect to with and without environment variables is given as under.

TABLE V. EFFORT ESTIMATION WITH AND WITHOUT ENVIRONMENT VARIABLES

Effort estimation without Environment Variables	Effort estimation with environment Variables
15.235	19.265
14.5262	18.562
16.235	20.256
15.9586	19.8568
14.2562	18.256

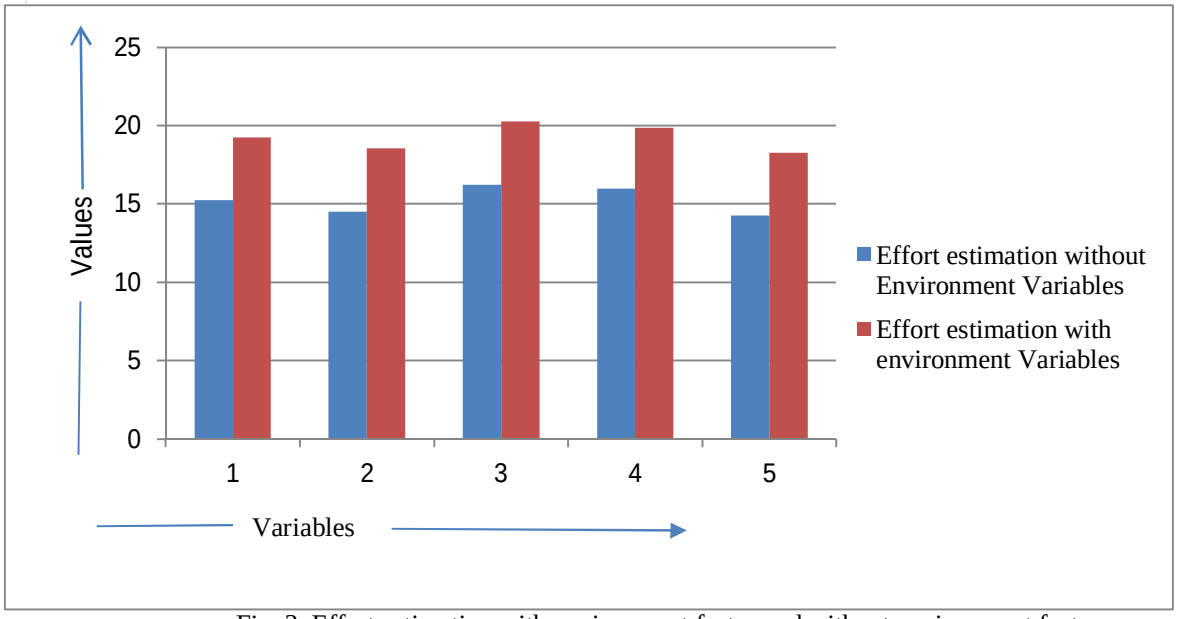


Fig. 3 Effort estimation with environment factor and without environment factor

Effort estimate is more accurate and prevent software crises since required time interval prediction is more as compared to exiting system

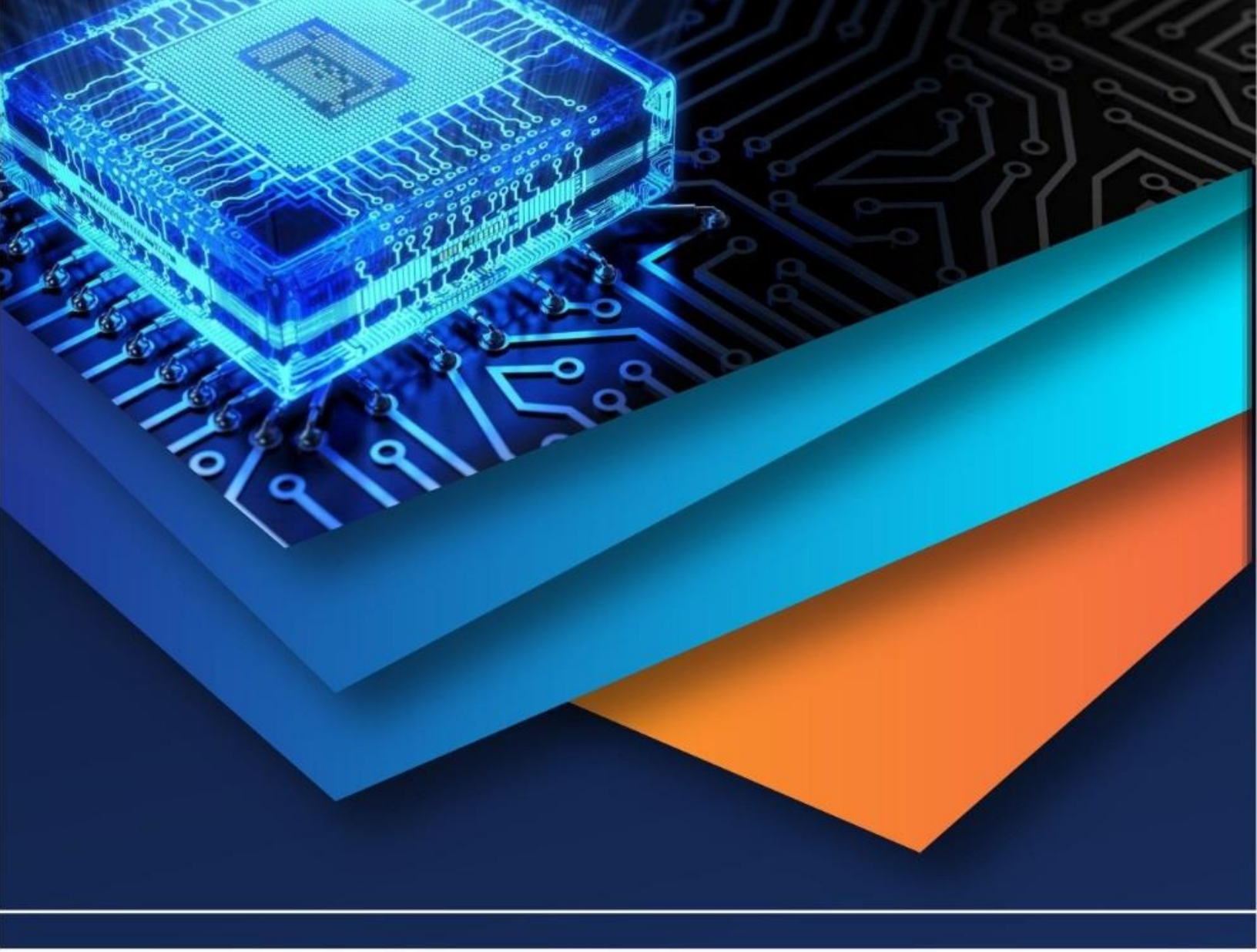
IV. CONCLUSION AND FUTURE WORK

This paper present comprehensive analysis of effort estimation mechanism which is required to save from software crisis. It is concluded that without anomaly consideration it is difficult to determine the accurate estimate of cost and effort. To determine actual effort and cost required for software development, anomalies and size both should be the factors. This will increase the time indicated at beginning required from software development but software crisis will be minimal by adopting this model. Implementation of the proposed system is conducted within MATLAB 2017a. The simulation result in the accurate effort estimation facility which is more than the system without considering the environmental variables. Environmental variables considered does not overload or increase the cost of overall project development but only enhances the estimation factor. Result is improved by the margin of 5%. In future, versatility of humans could also be considered for software effort estimation. This can play critical role in minimizing software crisis. Software crisis although can be minimized by considering environmental variables but still versatility factor can be introduced to achieve high degree of accuracy from the considered system. The multi heuristic approach can be used to achieve optimization within the software development process. Multi heuristic approach such as genetic approach along with versatility detection could be considered as future work.

REFERENCES

[1] W. D. Yu, A. Barshefsky, and S. T. Huang, "♦ An Empirical Study of Software Faults Preventable at a Personal Level in a Very Large Software Development Environment," IEEE Access, pp. 221–232, 2000.
[2] A. F. Sheta and S. Aljahdali, "Software Effort Estimation Inspired by COCOMO and FP Models : A Fuzzy Logic Approach," IEEE Access, vol. 4, no. 11, 2013.
[3] Md. Faiyazuddin (Ph.D., "Existing technqiues of software development," IEEE Access, no. 2012, pp. 40–72, 2011.
[4] E. E. Ogheneovo, "Software Dysfunction : Why Do Software Fail ?," no. April, pp. 25–35, 2014.
[5] F. Li, Z. Li, W. Huo, and X. Feng, "Locating Software Faults Based on Minimum Debugging Frontier Set," IEEE Access, pp. 1–18, 2016
[6] S. Valizadeh, M. Goodini, S. Ehsani, H. Mohseni, and F. Azimi, "Designing of a Computer Software for Detection of Approximal Caries in Posterior Teeth," vol. 12, no. 4, 2015
[7] C. Thirumalai, S. R.R., and R. R. L, "An Assessment of Halstead and COCOMO Model for Effort Estimation," Int. Conf. Innov. Power Adv. Comput. Technol. [i-PACT2017], no. April, pp. 1–4, 2017.
[8] N. Gupta, "Optimizing Intermediate COCOMO Model Using BAT Algorithm," ieee, 2014.
[9] M. Madheswaran and D. Sivakumar, "Enhancement of Prediction Accuracy in Cocomo Model for Software Project Using Neural Network," Ieee, 2014
[10] Jun Liu, Zheng Xu, Jianzhong Qiao, and Shukuan Lin, "A defect prediction model for software based on service oriented architecture using EXPERT COCOMO," 2009 Chinese Control Decis. Conf., pp. 2591–2594, 2009.
[11] N. Rutar and J. K. Hollingsworth, "Software Analysis Techniques to Approximate Data Centric Direct Measurements.," First Int. Work. High-performance Infrastruct. Scalable Tools, 2011.

[12] W. D. Yu, A. Barshefsky, and S. T. Huang, "♦ An Empirical Study of Software Faults Preventable at a Personal Level in a Very Large Software Development Environment," IEEE Access, pp. 221–232, 2000.
[13] A. F. Sheta and S. Aljahdali, "Software Effort Estimation Inspired by COCOMO and FP Models : A Fuzzy Logic Approach," IEEE Access, vol. 4, no. 11, 2013
[14] Md. Faiyazuddin (Ph.D., "Existing technqiues of software development," IEEE Access, no. 2012, pp. 40–72, 2011
[15] E. E. Ogheneovo, "Software Dysfunction : Why Do Software Fail ?," no. April, pp. 25–35, 2014.
[16] F. Li, Z. Li, W. Huo, and X. Feng, "Locating Software Faults Based on Minimum Debugging Frontier Set," IEEE Access, pp. 1–18, 2016
[17] S. Valizadeh, M. Goodini, S. Ehsani, H. Mohseni, and F. Azimi, "Designing of a Computer Software for Detection of Approximal Caries in Posterior Teeth," vol. 12, no. 4, 2015.
[18] C. Thirumalai, S. R.R., and R. R. L, "An Assessment of Halstead and COCOMO Model for Effort Estimation," Int. Conf. Innov. Power Adv. Comput. Technol. [i-PACT2017], no. April, pp. 1–4, 2017
[19] N. Gupta, "Optimizing Intermediate COCOMO Model Using BAT Algorithm," ieee, 2014.
[20] M. Madheswaran and D. Sivakumar, "Enhancement of Prediction Accuracy in Cocomo Model for Software Project Using Neural Network," Ieee, 2014
[21] S. Wu and I. Kuan, "Factors on Software Effort Estimation," Int. J. Softw. Eng. Appl., vol. 8, no. 1, 2017.
[22] Jun Liu, Zheng Xu, Jianzhong Qiao, and Shukuan Lin, "A defect prediction model for software based on service oriented architecture using EXPERT COCOMO," 2009 Chinese Control Decis. Conf., pp. 2591–2594, 2009
[23] N. Rutar and J. K. Hollingsworth, "Software Analysis Techniques to Approximate Data Centric Direct Measurements.," First Int. Work. High-performance Infrastruct. Scalable Tools, 2011.
[24] G. Mathew, T. Menzies, and J. Hihn, "Impacts of Bad ESP (Early Size Predictions) on Software Effort Estimation," 2016.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY