Software Reliability Estimation Using COCOMO II & Neuro Fuzzy Method

Rohit Malik[1], Kamna Solanki[2], Amita Dhankhar[3], Sandeep Dalal[4]
[1] M. Tech. Scholar, Software Engineering, U.I.E.T, MDU, Rohtak
[2] Assistant Professor, H.O.D of CSE Department, U.I.E.T, MDU, Rohtak
[3] Assistant Professor, U.I.E.T, MDU, Rohtak
[4] Assistant Professor, D.C.S.A, MDU, Rohtak

Abstract: Software Reliability Estimation is an essential part of the project management which affects the success of the project. Imprecise and undependable estimation will lead to failure of the project so it should be done carefully. It is the process of estimating the most realistic measure of effort required to produce a software. It alters from hardware reliability as it reflects the architecture integrity comparatively than composition integrity. This paper illustrates the reliability estimation of the software product which will be measured by various critical parameters. It is a pre-estimation technique in which tests are performed on various factors to check the reliability of the software as it is beneficial for the development stage. Techniques implemented for the reliability estimation are COCOMO 2 model which is an earlier methodology to check reliability of the software product whereas neuro fuzzy method and data sets are emerging techniques in current scenario. There is no single method that can be termed as the best methodology but there are mixture of techniques which can be implemented for the estimation of the software product and provide best-fitted results.

***Index Terms* - COCOMO 2, Estimation, Neuro-Fuzzy, Reliability.**

I. INTRODUCTION

Software engineering is an art related to evolution of software product by utilizing well defined technical standards, techniques and procedures which assures that the software is furnished on time and within cost frame along with the documentation that provides narration to the customer on how to utilize the software. It requires best quality of investigation and concentrated effort and administration of the two. With expanding complexity of the software, its development has become a more crucial mechanism which needs to be taken care off.

Impressive software project estimation is one of the ultimate challenge and a decisive action in the evolution of a software. Convenient project planning and management is not possible without a remarkable estimation. As a whole, the software business does not valuate projects and utilize estimates accordingly. This paper will undergo major circumstances which cannot be predicted so there is need to spotlight that how situation can be enhanced.

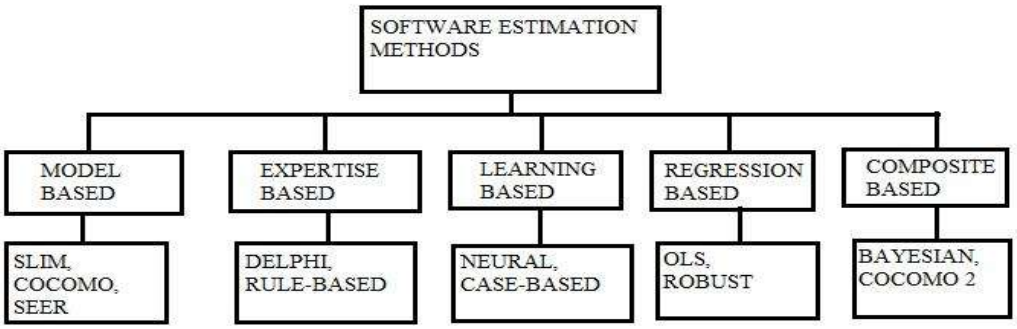
Qualities of a well-built software are:

i. Efficiency – When a software utilizes minimal accessible assets in the most proficient style and contribute suitable feedback in each case.
ii. Functionality – The software system will carry out the integral assignment entirely as mandatory by the customer for which it is refined.
iii. Maintainability - This attribute is important for both developer and user. The designer has to acknowledge quickly if there is any modification in the customer demand. Modification should be executed in such a way which will not affect the inclusive integrity of the software.
iv. Portability - Software must be accessible from any location. User can utilize it from any system.
v. Reliability - It is the probability of deficiency-freebie software application for a stated span of time within a described environment.

Software Estimation:

Administration of any project initiates with an efficient evaluation which is the foundation of the development stage of any project. Improper estimation will outcome the failure in the evolution phase. Elemental software estimation are cost, effort, schedule and size.

Estimation Methods in Software Engineering:



This paper will consider COCOMO 2 model for cost and effort estimation, integrates all three models of COCOMO 2 which are application composition, early design and post architecture model. However for size estimation this paper will make use of FPA (Function Point Analysis).

There are three models of COCOMO II explained as follows:

i. Application Composition – This model is suitable for those applications which can be fabricated by merging pre-packaged outcomes but can't be developed by application developers. This model will utilize object points methodology for size valuation. It will measure size of any tool on the basis of record and 3GL elements. Example - GUI builders, Query browsers, Database managers etc.

ii. Early Design – This model can be utilized for application creators, structure consolidation and framework growth segments. It utilizes unadjusted function points for the size assessment.

iii. Post Architecture – This model has identical methodology as of COCOMO 81 and utilizes entirely 17 cost drivers for software valuation and utilizes unadjusted function point and source lines of code for size valuation.

FPA (Function Point Analysis):

It will split the system into minor fragments so that complexity present in the system framework will become more noticeable and reviewed in a more efficient manner. FPA evaluates the size frame of the software on the justification of the functionalities to be outfitted by the software in terms of function count points which can be reformed into SLOC clearly if the identical SLOC for unit FPC is accessible.

Problems:

i. It is impossible to predict the unknown – Every software framework is exceptional even if the demands are equipped, it is difficult to evaluate a complicated system that is going to be reinforced with the granted requirements.

ii. Estimates are generally positive – There are two perceptions either the developer think that they are more constructive or they do not recognize the complexity of the software.

iii. Estimated time is evermore utilized – If time frame of two days is granted to the developer to finish the task , he will utilize full frame of time however he don't think to develop the task early.

iv. Estimates do not deliberate efficient divergences between programmers – It is almost unacceptable to evaluate precisely when developer has two un-identified variables like speed for expansion and measure of work.

v. Variations in demands do not reflect on valuations – Almost every project have changes in demands all over project, but the estimation will remain same as no modifications are allowed. Each attribute which is supplement to the project have an influence on the estimates otherwise this entire technique of estimation is a prank.

vi. Estimates are not treated as estimates - There is a confusion in the correct meaning of the term estimate. There must be not too many plans which are dependent on estimates. If the product which have estimates are in-turn dependent on another product then it will create a chain of delay, by which estimation cannot be achieved.

II. METHODOLGY

This paper will consider three different company with their values like line of code, function points and actual efforts and by applying formulae, predicted value of the estimation is achieved. This company data is collected from google open source and perform operations with the help of COCOMO 2 model and Fuzzy tools set.

Example set:

COMPANY NAME	C1
LINE OF CODE	14000
FUNCTION POINTS	218
ACTUAL EFFORTS	38

COMPANY NAME	C2
LINE OF CODE	12000
FUNCTION POINTS	187
ACTUAL EFFORTS	32

COMPANY NAME	C3
LINE OF CODE	9000
FUNCTION POINTS	140
ACTUAL EFFORTS	24

COMPANY NAME	COCOMO PREDICTED ESTIMATION
C1	33.54
C2	27.89
C3	18.32

COCOMO 2

Various formula are used in the COCOMO 2 model are-
Estimated effort per months = A * (size) E where,
Value (constant set) = 2.94
Size = KLOC provided by the company
E = Estimated effort based on the 17 effort multipliers which are grouped in 4 category which are explained as follows –

i. Product Attributes – Software Reusability, Database, and complexity of the product.

ii. Computer Attributes – Execution time constant, Storage and Total predicted time to complete software.

iii. Personal Attributes – Tool cost, Programmer capability and Analyst capability.

iv. Project Attributes – Development time and Developed version of software tool cost.

Following programming languages are used to get the result:

LANGUAGE	SLOC/UFP
C	128
C++	29
JAVA	46

Post Architecture Counterpart Drivers –

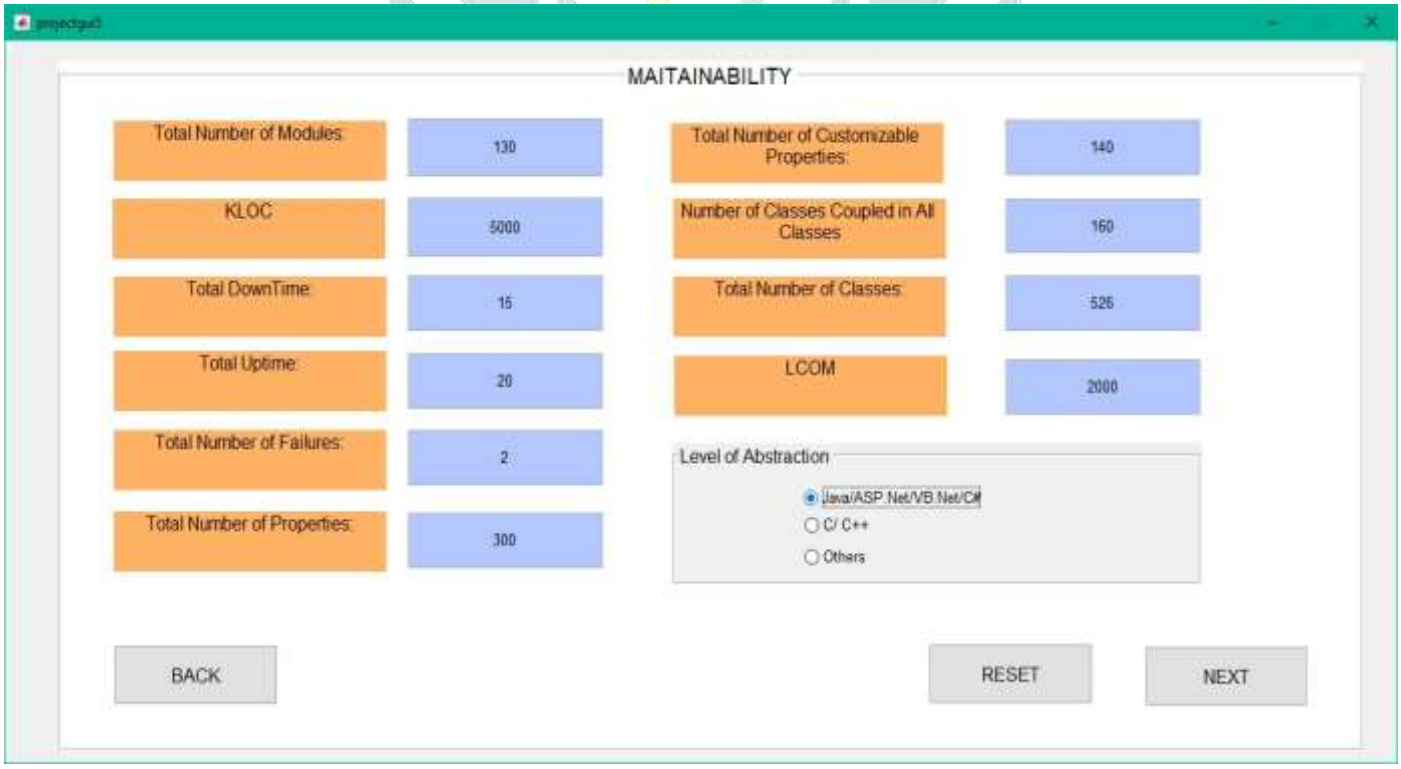
RELY, DATA, CPLX, DOCU, RUSE, TIME, STOR, PVOL, ACAP, PCAP, PCON, AEXP, PEXP, LTEX, TOOL, SITE AND SCED.

Formula used in the software reliability estimation –

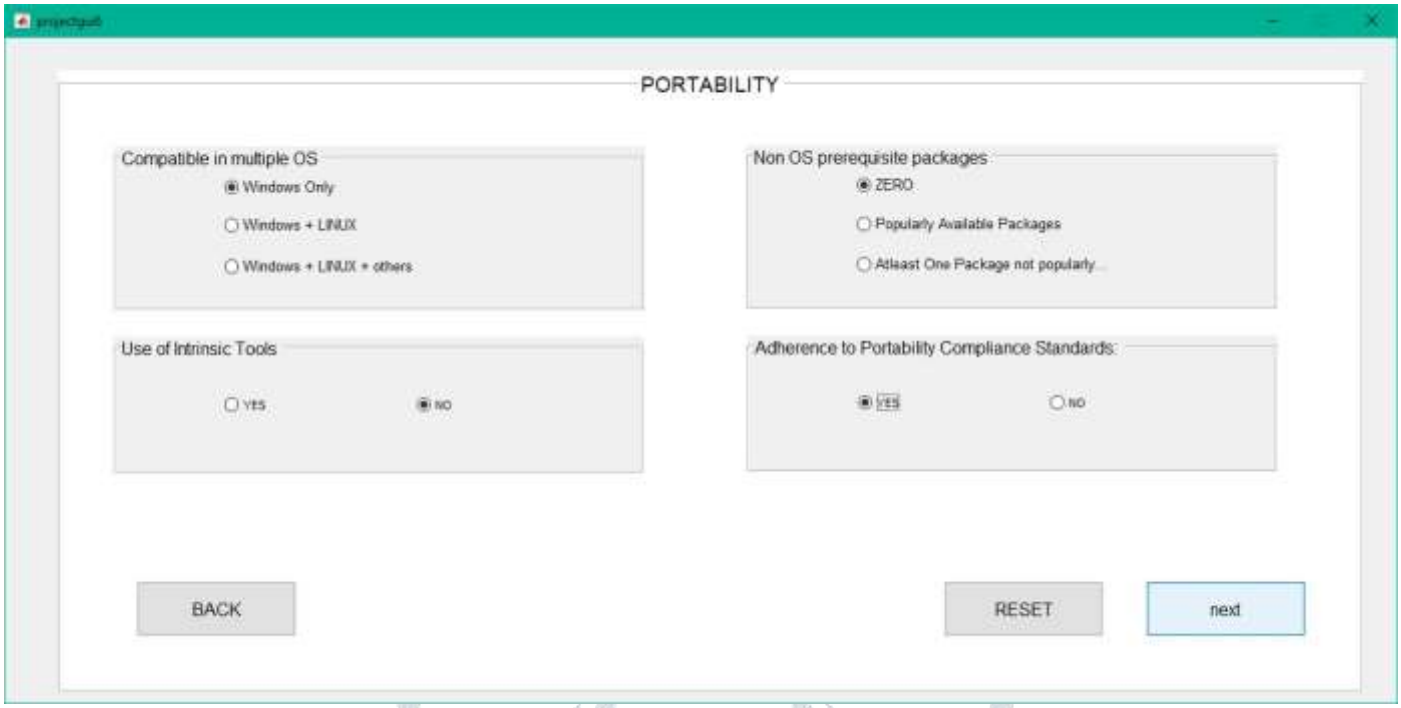
i. Unadjusted Function Points = ∑ (External Input) + ∑ (External Output) + ∑ (External File) + ∑ (External Inquiries) + ∑ (Internal File).
ii. Degree of Influence= ∑ General Application Characteristics [i] where i = 1 to 14.
iii. Technical Complexity Factor = (0.65 + 0.01 * Degree of Influence).
iv. Function Points = (Unadjusted Function Points * Technical Complexity Factor)/100.
v. Person Month = New Object Points / Developer experience and capability.

III. RESULTS AND DISCUSSION

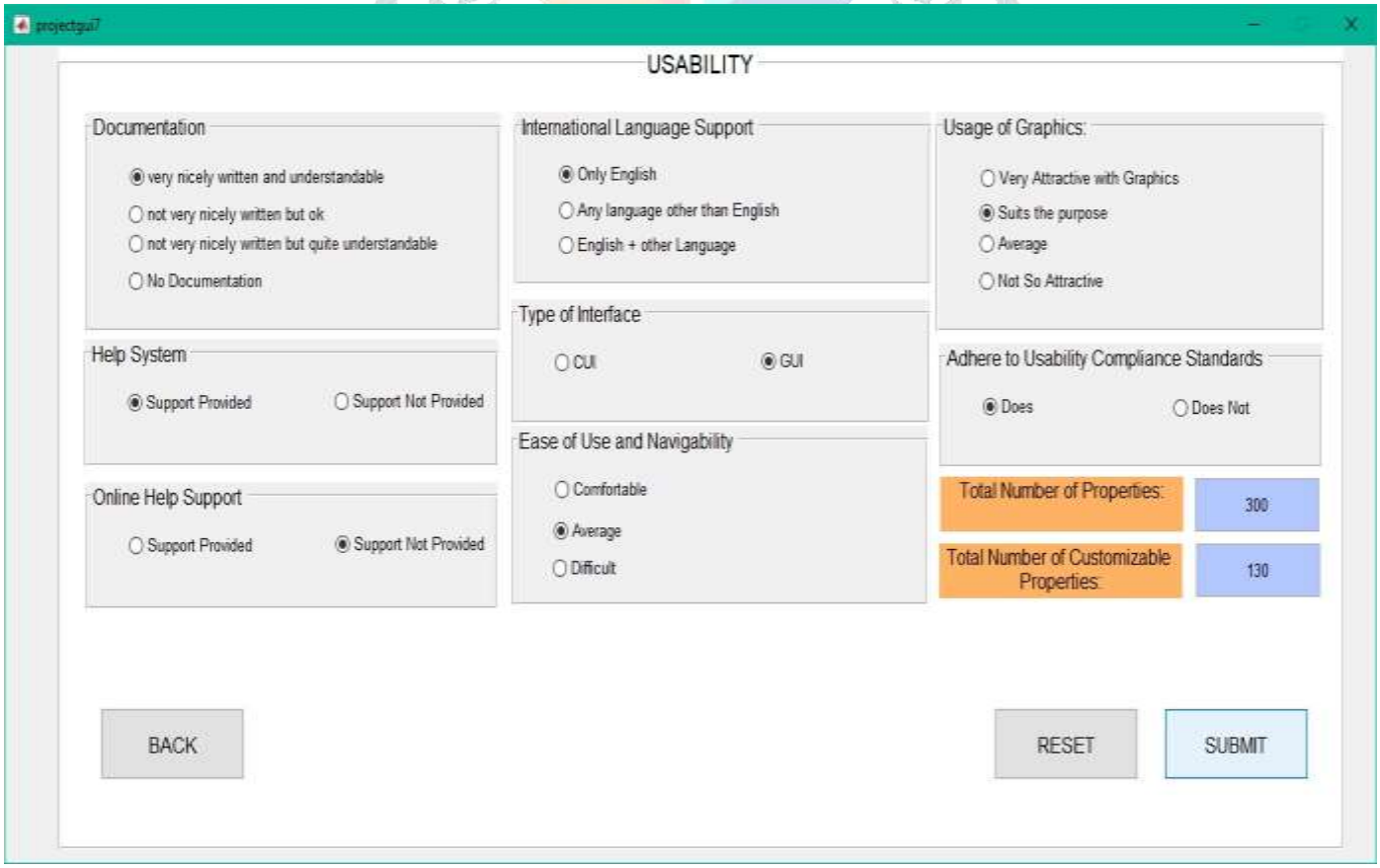
In "Maintainability" attribute this paper will consider total number of modules, KLOC, total down and up time, total number of failures, total number of customizable properties, number of coupled classes, total number of classes, LCOM and level of abstraction.



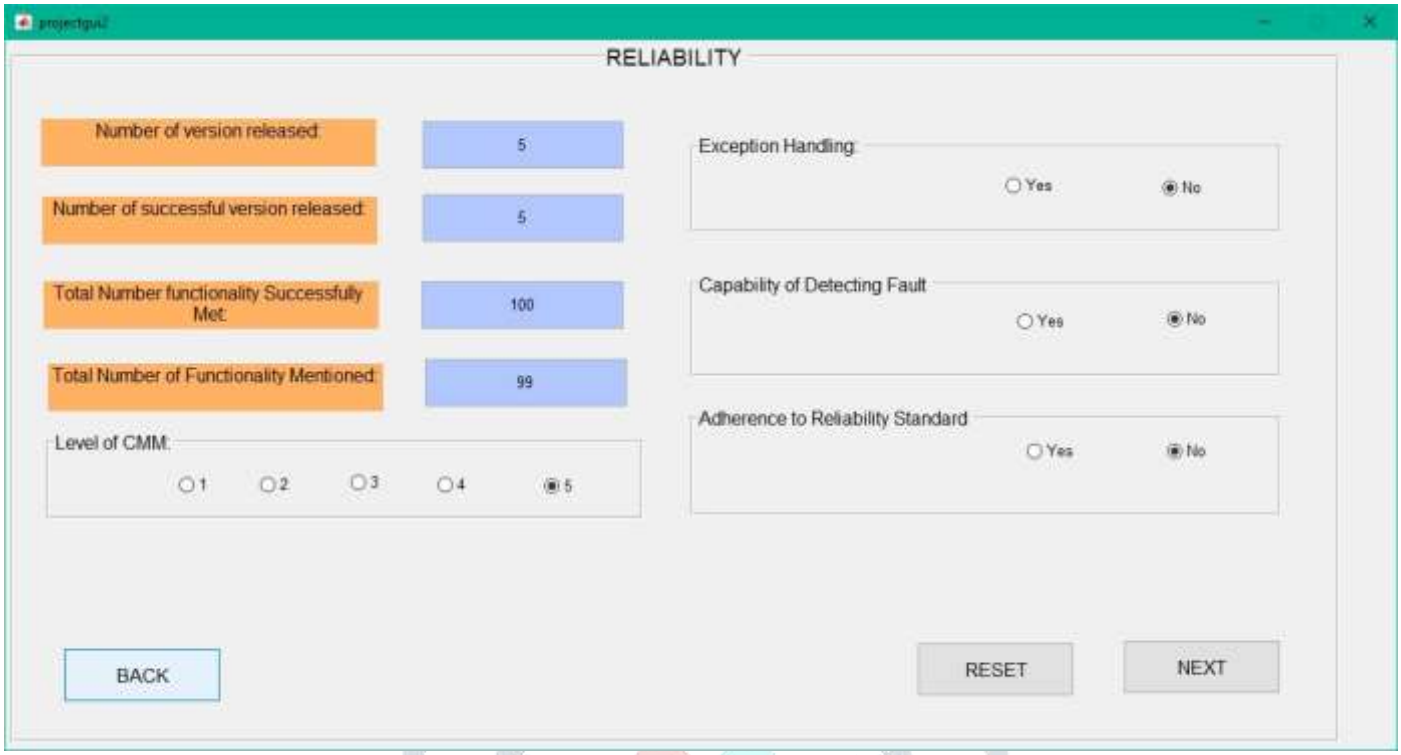
In "Portability" attribute this paper will consider compatibility in multiple Operating system, use of intrinsic tools, Non-OS prerequisite packages and adherence to portability compliance standards.



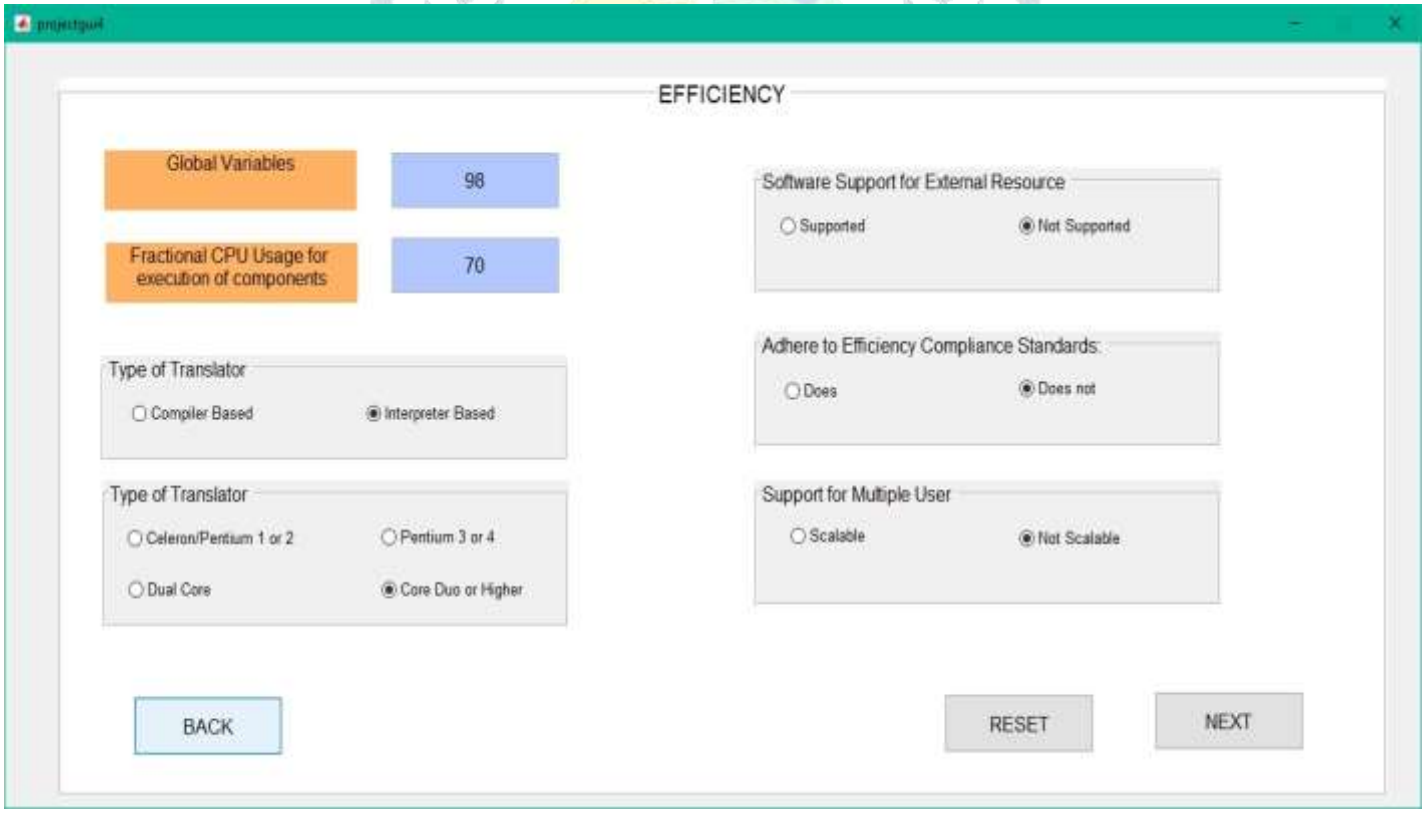
In "Usability" attribute this paper will consider documentation, help system, online help support, international language support, type of interface, ease of use and navigability, use of graphics, adherences to usability compliance standards and total number of properties and total number of complicated properties.



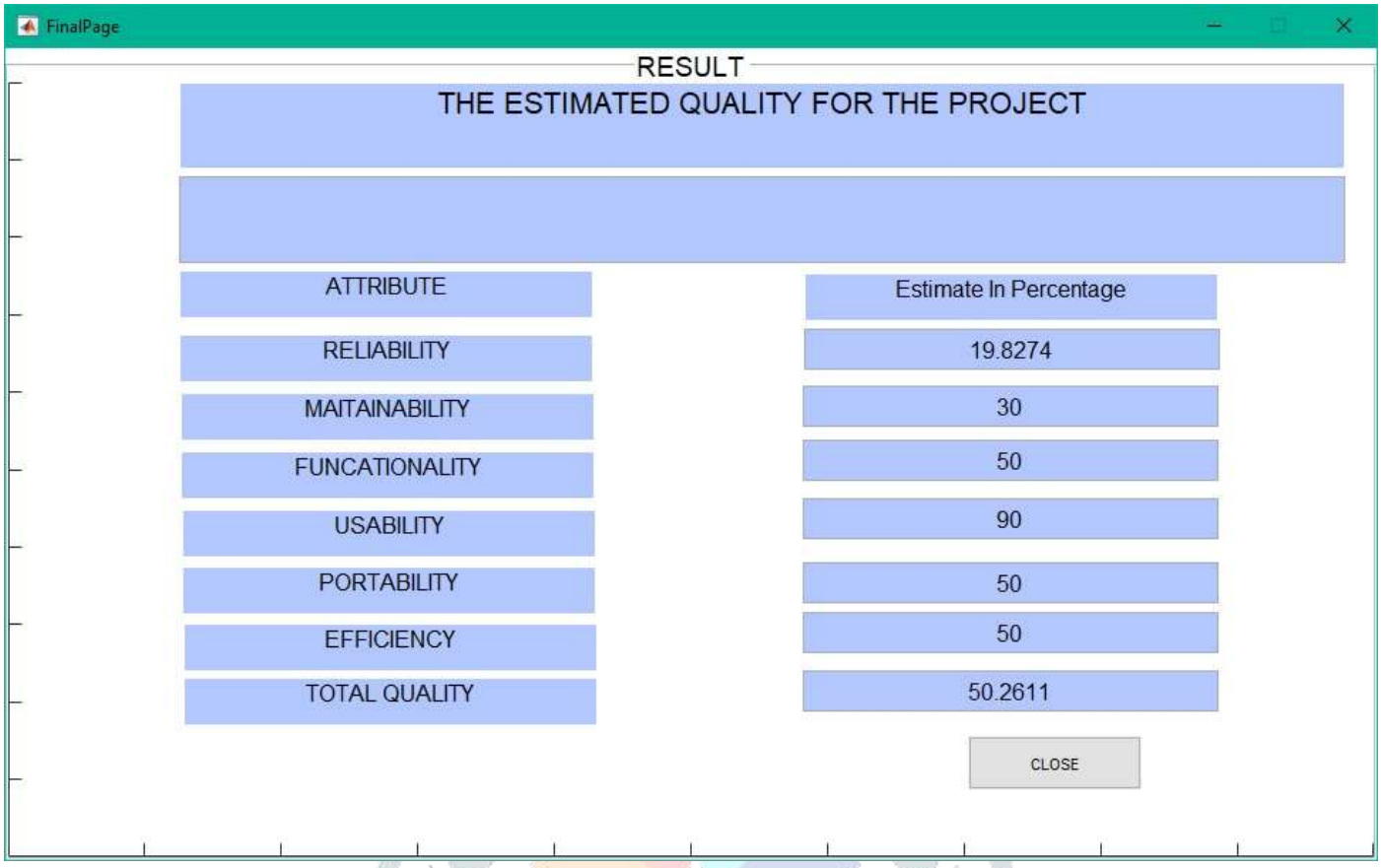
In "Reliability" attribute this paper will consider number of released versions, number of successful version released, total number of functionality which is successfully met, total number of functionality mentioned, exception handling, capability of detecting fault and adherence to reliability standard.



In "Efficiency" attribute this paper will consider global variables, fractional CPU usage for execution of components, type of translator, software support for external resource, adherence to efficiency compliance standard and support for multiple users.



Final Result of the estimates quality of the project, here total quality will be around 50.26 % performed on various reliability factors.



IV. CONCLUSION

From the discussion this paper conclude that a lot of work is accomplished in the area of software reliability estimation and planning. It is an important perspective to achieve best-fitted accuracy. Various techniques like COCOMO 2 model, neuro fuzzy method and data sets are utilized for the better reliability estimation. This paper describes that various parameters are considered like reliability, efficiency etc. on which operations are performed by passing data set values to check out the reliability in a pre-developed stage. Parameters like function points, KLOC and actual efforts of various companies are compared and reliability estimation is predicted. This paper illustrates that 17 cost drivers are used along with various programming languages with their UFP values. Data sets are provided by the owner of the company on which tester will perform operations to decide the reliability estimation. Finally after performing data set operations on different parameters it will provide result in measure of the overall quality of the software product.

REFERENCES

[1] Rathi.J, Kamalraj.R and Karthik.S, "Survey on Effective Software Effort Estimation Techniques", International Journal of Advanced Research in Computer Engineering & Technology, vol. 1, 2012.

[2] Kavita Choudhary, "Parametric Estimation of Software Systems", International Journal of Soft Computing and Engineering, ISSN: 2231-2307, vol. 1, issue-2, 2011.

[3] R Raja Ramesh Merugu and Venkat Ravi Kumar Dammu, "Effort Estimation of Software Project", International Journal of Advanced Research in Computer Engineering & Technology, ISSN: 2278 – 1323, vol. 1, issue-10, 2012.

[4] Jovan Zivadinovic, Zorica Medic, Dragan Maksimovic, Aleksanda Damnjanovic and Slanana Vujcic, "Methods of Effort Estimation in Software Engineering", International Symposium Engineering Management And Competitiveness 2011 (EMC2011) June 24-25, 2011, Zrenjanin, Serbia.

[5] P.K. Suri and Pallavi Ranjan, "Comparative Analysis of Software Effort Estimation Techniques", International Journal of Computer Applications, doi: 10.5120/7479-0540, vol. 48, issue-21, 2012.

[6] Suresh Joseph K. and Ravi Chandran T, "Genetic approach for Software Effort Estimation: A Variant for COCOMO", International Journal of Computer Applications, ISSN: 2278 -7720, vol. 1, issue-3, pp. 58 – 64, 2012.

[7] Narsing Rao and CH. Mahesh Reddy, "Predictive and Stochastic Methodology for Software Effort Estimation", International Journal of Advance Research, Ideas and Innovations in Technology, ISSN: 2454-132X, vol. 3, issue-6, 2017.

[8] Elham Khatibi and Roliana Ibrahim, "Efficient Indicators to Evaluate the Status of Software Development Effort Estimation inside the Organizations", International Journal of Managing Information Technology, doi: 10.5121/ijmit.2012.4303, vol. 4, issue-3, 2012.

[9] Ali Bou Nassif, Luiz Fernando Capretz and Danny Ho, "Enhancing Use Case Point Estimation Method Using Soft Computing Techniques", Journal of Global Research in Computer Science, ISSN: 2229-371X, vol. 1, issue-4, 2010.

[10] Jyoti G. Borade and Vikas R. Khalkar, "Software Project Effort and Cost Estimation Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277-128X, vol. 3, issue- 8, pp. 730-739, 2013.

[11] Chetan Nagar and Anurag Dixit, "Software Efforts and Cost Estimation with a Systematic Approach", Journal of Emerging Trends in Computing and Information Sciences, ISSN: 2079-8407, vol. 2, issue-7, 2011.

[12] S.K. Mohanty and A.K. Bisoi, "Software Effort Estimation Approaches - A Review", International Journal of Internet Computing, ISSN: 2231 –6965, vol. 1, issue-3, 2012.