International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
Volume 3 Issue 1 –NOVEMBER 2013.

# Developing a Reliability Prediction System Using Multivariate Analysis Theory on Software Quality Metrics

E. Umamaheswari [#1], Dr.D.K.Ghosh [*2]

[#] *School of Computing Science and Engineering, VIT University, Chennai, TN, India.*

[1] umamaheswari.e@vit.ac.in

[*] *Professor, VSB Engineering College, Karur, TN, India.*

***Abstract-*** **Software Reliability Prediction techniques are Along with the existing 30 software measures, nine more measures are identified which results in the best performance of the software application. Predicting the reliability of software systems, failure data should be measured by different ways during the development process and its execution phases. This paper proposes a structure of reliability prediction which will be used to rank the software measures based on the structure with multivariate analysis theory .This structure will be accurate with the experimental data obtained, and it also checks that the software has met its reliability or not and if not, it will continue the process again to achieve its reliability.**

***Index Terms-*** **Software Reliability Prediction, Software measures, multivariate analysis theory.**

## I. Introduction

RPS (Reliability Prediction System) is an outline for predicting the reliability of software by taking the measures of software. Reliability Prediction can be achieved by applying multivariate analysis theory (analysis of data involving more than one variable). Ranking of existing software was done before by taking the opinions of the 30 experts and in this report it is possible by the help of the RPS framework to rank the software methods. There are few measures included and explained in this paper which helps more efficiently to calculate the reliability of the software.

There are four types of models which have been considered as potential candidates for modeling the reliability of software. These include reliability growth models, input domain models, architectural models and Beginning prediction models. (1)Reliability growth model captures failure performance during testing and generalizes its performance during procedure. Hence this category of models uses failure data and observes the failure data to derive reliability predictions. (2)Input Domain model uses properties of the input field of the software to derive correctness which approximates from test cases that executed properly. (3)Architectural models stress on the architecture of the software and derive reliability estimates by combining estimates obtained for the different modules of the software. (4) Beginning prediction model uses characteristics of the software development process from requirements to test and estimates this information to performance during operation.

### A. *Reliability*

Reliability is probability of the non-occurrences of error. It states that an item will perform a defined method without failure during certain period of time. The numerical values of the reliability is expressed as a probability from 0 to 1and it has no units [1].Reliability is one of the validation criteria for measuring and ranking software among correlation, consistency, tracking, predictability and discriminative power. System reliability and accessibility are precise as a part of the non-functional requirements for the system. It is very important to choose an appropriate metric to specify the overall software reliability. It gives measurement by input software data and outputs a single numerical value.

### *B. Reliability Prediction System*

Reliability prediction system describes the process which is used to estimate the constant failure rate during the useful life of software. This is one of the general forms of reliability analysis. Reliability prediction system predicts the failure rate of components and overall software reliability. These prediction system are used to calculate approximately design feasibility, evaluate design alternatives, identify possible failure areas, trade-off system design factors, and tracks reliability enhancement [2]. The impact of future proposed design of software changes on reliability is determined by comparing the reliability predictions of the existing and proposed designs of the software. The capability of the design of software to maintain an acceptable reliability level can be accessed through reliability predictions.

### *C. Multivariate Analysis Theory*

Multivariate analysis theory consists of a set of methods that can be used when numerous measurements are made on each individual or object in one or more sample [4]. With univariate analysis, there is only one dependent variable of interest but by using multivariate analysis theory there are more than one variable involved in analysis of data. By using this theory richer realistic design of the software will be obtained. It also helps to predict the reliability and determine structure of the software.

The ranking of any software measure is predicted

On the following values:

1. The value of 1 is assign to best likely situation and hence it represents the highest reliability of any measure of the software.

2. The value of 0 is assign to worst situation and has the lowest possible reliability of any measure of the software.

3. The ranking according multivariate analysis theory can be done by predicting values lying between the first and the last ranking criteria levels which take values between 0 and 1.

Values to be selected depends on the relative effectiveness of the ranking criteria level considered.

### . *D. Software Quality Metrics*

Software metrics is a measure of property of a piece of software or its specifications [3]. It is a quantitative measure of degree to which a system component or process have a given attribute (i.e. guess about a given attribute). There are three main categories in which metrics are classified. They are:

**Process metrics:**

This metrics deals with the activities which are related to production of software. It is mainly concerned to improve the process efficiency of the SLC.

**Project metrics:**

This metrics deals with more relevant to project team for developing software. It can be used to measure the efficiency of a project team or any other tools being used by team measures. It requires hardware, people and knowledge to measure its attribute.

**Product metrics:**

This metrics deals with the explicit results of software development activities. This requires deliverables, documentation of products used in the approach of the software product being developed.

## II. BACKGROUND

To Determine Reliability Objective Step 1 To Carry Out Software Testing Step 2 Failure Data Collection Step 3 To Apply Software Reliability Tools Step 4 Selection of Appropriate Software Reliability Models Step 5 Use of Software Reliability Models to Calculate Current Reliability Step 6 Start to Deploy Step 7 To Validate Reliability in the Field Step 8 Feedback to Next Release Step 9 Is Reliability Objective met? Yes No Continue with the Testing

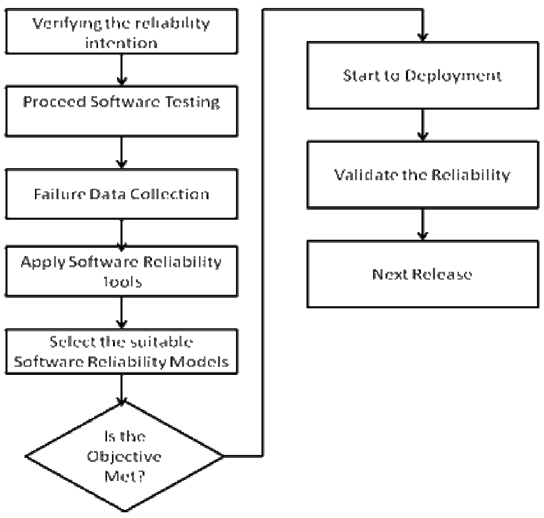

Fig 1: Reliability Prediction Framework

## III. METHODS

There are already 30 measures based on the IEEE standard according to which reliability prediction can be performed through which the high quality of the software can be achieved. These measures are listed below: [5]

1.) Bugs per line of code
2.) Cause and effect graphing
3.) Code defect density
4.) Cohesion
5.) Completeness
6.) Cumulative failure profile
7.) Cycloramic complexity
8.) Data flow complexity
9.) Design defect density
10.) Error distribution
11.) Failure rate
12.) Fault density
13.) Fault-days number
14.) Feature point analysis
15.) Function point analysis
16.) Functional test coverage
17.) Graph-theoretic static architecture complexity
18.) Man hours per major defect detected
19.) Mean time to failure
20.) Minimal unit test determination
21.) Modular test coverage
22.) Mutation testing (error sending)
23.) Number of faults remaining (error sending)
24.) Requirements compliance
25.) Requirements specification change requests
26.) Requirement traceability
27.) Reviews, inspections and walkthroughs
28.) Software capability maturity model

29.) System design complexity

30.) Test coverage

There are some software measures excluding above measures which can be added for the high reliability performance which are listed and explained below:

CLASS COUPLING:

This software measure refers to the degree of interdependence parts of design of the software. This measure basically couples between object classes i.e, it scans the design of classes that how one class relates to other classes. It is defined as the total summation of the all the classes to which a class is coupled. Where ever there is dependency on any one functionality on other functionality of the software, then by using this measure we can rank the reliability of the software.

APPLICATION LEVEL CLASS HIERARCHY NESTING :

This software measure assesses how many classes affects application level class. Inheritance concept is used here to know the depth of the tree structure which is helpful to know the performance of the software. Large number used in tree means if nesting in the program is more it leads to the design problem of the software. So by this measure reliability of the design of the software can be checked.

FACTOR COVERAGE:

This software measure measures the capability of the software to automatically recover from the unwanted failure during execution. Coverage is defined as the probability that the software recover from breakdown.

Markov chain model is used in this measure by which software can predict coverage before its failure and make software reliable from everlasting, irregular, and temporary errors.

LACK OF COHESION IN METHODS (LCOM):

This software measure is a comparative pointer of cohesion of a class. This is a comparison between the number of correlated methods and the number of irrelevant methods from a design perspective to check whether there is any instance variable is shared between them. The LCOM provides a measure of the relative dissimilar nature of modules in software. The small number of modules implies greater similarity of features and therefore it measures the attributes of software.

NUMBER OF CHILDRENS (NOC):

This software measure counts the sub-modules of the software being measured and it measures the complexity of the software. The greater the number of sub modules, the greater the possibility of inappropriate generalization of the all the classes. If any software has a large number of modules embedded in it, it may be a case of exploitation of the software. Therefore this measure checks the hierarchy of the modules and sub modules of the software for the better performance.

NUMBER OF CLASS METHODS IN AN APPLICATION:

This software measure measures the size of the software from the number of methods in a program.

It indicates poor design of the system if the services are handled by the class itself.

The number of methods accessible to the class affects the size of the class. Implementation of the methods in a class can be done as follows:

All the methods will be identified inside the class while measurement.

Number of the methods will be counted on methods retrieved in step 1.

This number is now the value of the measure of number of class methods in a class.

KEY CLASSES APPLICATION:

This software measure estimates the number of key classes in a system. The value of this measure is a pointer which required developing the system. Key classes can be the mid points of reprocess on future projects, since they are expected to be needed in other domains in the production. The number of key classes is a sign of the volume of work needed in order to develop software. Therefore this measure helps to develop the software without the class and it is reliable hence this type of software can be long term reusable.

MUTATION TEST SCORE:

Mutation is a single-point; syntactically accurate transform, introduced in the program to be tested. This software measure is designed for the purpose of providing a measure of the effectiveness of a testing data set. A high score indicate that data set is very efficient for the program with respect to mutation fault coverage. The mutation score is the ratio of the non-equivalent mutants of Program (which are clear from Program for at least one data point of the input domain) which are killed by a test data set.

Equivalent mutants are mutant programs that are functionally equivalent to the original program and therefore cannot be killed by any test case .A set of mutants of Program consists of a set of programs which differ from Program in containing one mutation from a given list of faults of the most likely faults introduced by programmers using the language of Programs.

Therefore this measures the reliability by finding test cases that kill all nonequivalent mutants.

WEIGHTED METHOD PER CLASS (WMC):

This software measure is the computation of weighted methods in a class. Every method inside the class is weighted by complexity metric and this weight is added up so as to arrive at Weighted Method per Class. The implementation of this metrics can be done in the following steps:

1. Check the method of the class and name the complexity of all statement in this method according to the mapping of weights.

2. The complexity of method is defined as the total of the results produced in step 1.

## IV. CONCLUSION

The aim of this paper was to propose a framework for Reliability Prediction system. The steps involved in the framework are illustrated and which operations are performed when is also determined. There were 30 software measures which are good indicators of the software reliability were elicited by the expert opinions. This paper proposes 9 more measures which will help now them to consider these measures and produce more reliable system. Based on the given framework a tool can be implemented which can take all inputs as experimental data and gives output accurate compare to the research done till now by taking expert opinions

This study is only the initial stage of a long-standing study in predicting software reliability. Future research includes the identification of the RPSs including all the features which this paper has proposed. Achievable functions that allow quantification of reliability from RPSs need to be investigated.

## V. REFERENCES

[1] M. King, B. Zhu, and S. Tang, "Optimal path planning," *Mobile Robots*, vol. 8, no. 2, pp. 520-531, March 2001.

[2] H. Simpson, *Dumb Robots*, 3rd ed., Springfield: UOS Press, 2004, pp.6-9.

[3] M. King and B. Zhu, "Gaming strategies," in Path Planning to the West, vol. II, S. Tang and M. King, Eds. Xian: Jiaoda Press, 1998, pp. 158-176.

[4] B. Simpson, et al, "Title of paper goes here if known," unpublished.

[5] J.-G. Lu, "Title of paper with only the first word capitalized," *J. Name Stand. Abbrev.*, in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Translated J. Magn. Japan*, vol. 2, pp. 740-741, August 1987 [*Digest 9th Annual Conf. Magnetics Japan*, p. 301, 1982].

[7] M. Young, *The Technical Writer's Handbook*, Mill Valley, CA: University Science, 1989.