RESEARCH ARTICLE OPEN ACCESS

Dynamic Metric For Enhancing Software Reliability And Testabiliy Using Genetic Algorithm

Vaid Singh[1], Charnpreet Kaur[2]
Research Scholar[1&2], Department of Computer Science and Engineering
RIMT, Fatehgarh Sahib
Punjab - India

ABSTRACT
The overall aim of the software industry is to make sure delivery of high quality software to the end user**.** Software metrics are necessary to measure quality in terms of software performance and reliability connected characteristics like dependencies, coupling and cohesion etc. It provides a way to measure the improvement of code during development and having direct relationship with cost and time incurred in the software design and development at their later stages. It is supportive in developing good quality software that must be built in cost-effective manner and within time constraints to met end user requirements. Software Quality depends upon Software metrics. Software metric is the measurement of some property of software Hence performance of software directly depends upon the type of metric used for its estimation. Static metrics used in past are becoming outdated due to the altering nature of software. Because static metric does not give exact results about the dynamic run time behaviour of the software and software are becoming extra composite and having multiple user interfaces today. This leads to the interest of developers tending toward dynamic metric. Dynamic Matrices give exact results about the dynamic dependencies that are real during the run time of software. Developers are proficient to get dynamic analysis of the system that results in better performance of system. When a system is improved with new features then Reliability and Testability of the system is measured using an innovative dynamic metrics helps in growing maintainability and Testing of the system. The system developed following evaluation will be more compatible, a smaller amount prone to errors and with improved performance.

Keywords:- **Quality, Dynamic, Static, Software**.

I. INTRODUCTION

1.1Software Quality: Software Quality is to calculate a process of method and components of a system meeting the necessities that are already specified. We can also say that in which it can assemble customers or users necessities also. Relatively a single factor, quality in software is best viewed as a trade off between a set of different goals. Explicit attention to uniqueness of software quality can lead to important savings in software life-cycle costs. Distinctiveness of good quality software includes the Understand ability, Completeness, Conciseness, Portability, Consistency, Maintainability, Testability, Usability, Reliability and Structuredness.
Efficiency: It follows two types of criteria.
Internal Criteria, in which it is not able to be seen to the user and it is code-dependent and is for developer only.
External Criteria which is an understanding in operational mode by users when running the software**.**

1.2 Software Metrics: Software metric is one of the significant aspects of software engineering acts as an indicator for software attribute. It plays a significant role in understanding the vital concepts in the field of software engineering Software Metrics can be defined by measuring property or characteristic or quality of a software objects connected to any great and composite software project. In a broader term, it is a quantity up to which a system object can grasp a exact quality or characteristics. Software metric are supportive in improving the quality of software, planning the budget, its cost estimation etc., Software metrics explains the activities connected with measurements in software engineering. The metrics are practical to software development process and the product so as to get the significant information.

1.3 Software metrics can be classified into the three categories:

Product metrics explain the characteristics of the product. These characteristics contain size, complexity, design features, performance, and quality level. Product quality metrics includes four types of metrics, mean time to failure, defect density, customer problems and customer satisfaction. The metrics mean time to breakdown and failing densities are the intrinsic product quality metrics. This measures the number of “bugs” or in other words number of “defects” in software. Customer problem metrics measures the amount of problems that are faced by the user when he operates the software. The customer approval metric works on five point scale to know satisfaction level of the user, that is, whether a customer is very satisfied, neutral, dissatisfied or very dissatisfied.

Process metrics can be used to get better software development and maintenance. Process quality metrics**,** comprises of metrics as failing density during machine testing, fault arrival pattern during machine testing, phase based defect removal pattern, defect removal effectiveness.

Project metrics describe the project uniqueness and execution. The intent of project metrics is twofold:
–to diminish the development schedule,
–to assess product quality on an ongoing basis and when necessary, amend the technical approach to recover quality.

II. STATIC AND DYNAMIC METRICS

Static Metrics: Static metrics are those which work ahead the code which is not running, that is, non-executable code. Whereas on the other hand, Dynamic metrics are those which performs study on the executable or in other words running code. In previous years, a variety of techniques have been developed which works to get the dynamic knowledge of a program. These techniques are like program slicing, run time languages, instruction counts etc. The objects that are measured using the software metrics are algorithms, program and executions.

Dynamic Metrics: Dynamic metrics helps to estimate the dynamic behaviour of a function at run-time. The outcome of dynamic measures are much additional accurate than the static measures. The complexity of the dynamic behaviour of real-time applications motivates a shift from static metrics to dynamic metrics. Dynamic metrics that can be used to estimate relevant runtime properties of programs, with the vital goal of establishing some standard metrics that could be used for quantitative analysis of standard programs. Dynamic Metrics are derivative from a study of code while it is executing. Thus, dynamic metrics can only be calculated on the software as it is executing. For example: extent of class usage, Dynamic Coupling, and Dynamic Lack of Cohesion.

Static Metric vs. Dynamic Metrics:

Objectives	**Dynamic**	**Static**
Accurate and Well Organized Result	More	Less
Connected	With executable code	With non-executable code
Independent of Input Data	No	Yes
Independent of running environment	No	Yes
Accuracy	More	Less

III. DYNAMIC MATRICES CLASSIFICATION

Cohesion: It refers to how very much and powerfully the modules relates to each other. It is expressed as high cohesion and low cohesion. The modules that have high cohesion are preferred additional over the modules with low cohesion. Dynamic cohesion compute is based upon the four types of relationships among elements, as follow:

- Write dependence between attributes and methods.
- Read dependence between methods and attributes.
- Call dependence between methods.
- Reference dependence between attributes.

Coupling: It shows the dependency of one program module on an additional module. It is expressed in the form of low coupling and high coupling. Low coupling is associated to high cohesion and vice-versa.

Inheritance: Inheritance metrics calculate various aspects of inheritance such as depth and breadth in a hierarchy and overriding complexity. The inheritance metrics provide us information in family member to the inheritance tree of the system. Inheritance is a key feature of the OO example. This mechanism supports the class hierarchy design and captures the IS-A relationship between a super class and its subclass.

Polymorphism: Polymorphism means using the comparable function additional than once. Polymorphism is a software feature that required to be calculated dynamically. This metric was examined mainly in the context of software reusability next to with the determining total quality of the system. Various metric are used for measuring polymorphism, which provides a purpose and precise mechanism to recognize and count the dynamic polymorphism.

IV. PROPOSED WORK

The present work is about increasing the performance of the dynamic metric by adding the factors. These factors help to increase the functionality of the software system. Now days, software quality metrics are used. These metrics are used to detect the quality of the various software products. Quality of the software product depends upon the features added. Suppose a product has the high features than the other one then it becomes the best quality product. But there are many other problems have to face. With this the problem of reliability occurs, as the product with high features declared as the best quality product, but nobody wanted to know either the features are reliable to that particular product or not. With the additional features the performance of the product is also effects. The performance of system decreases. Hence the need of testability is also required here.

V. METHODOLOGY

In the proposed methodology, the research work contributes to design a new metrics by adding some factors in the existing metrics. These factors help to enhance the performance of dynamic metrics. These factors are testability and reliability. The following figure shows the flow in which the research work will proceed.

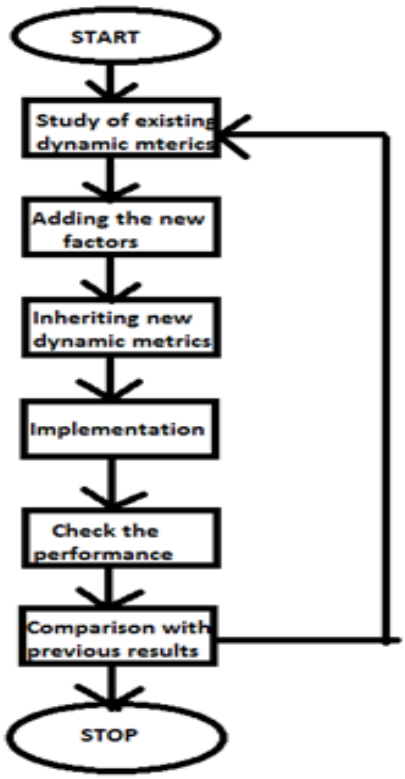

Figure 3.1: Procedure for the Research Work

Testability: Testability is necessary in the Dynamic metrics. It is the non-functional requirement. It defines the belongings of measuring the effortlessness of testing. Testability allows the component to be tested in separation. When the testability takes place in the system, the customer reports the smallest amount number of defects. The testable products are simple and the cost to continue product is also fewer. Testability is also significant for the maintainability of software product. When software is tested, firstly a piece of code is tested. The errors are established in that piece of code. After that the whole system is tested. Hence testability increases the maintainability of the system.

Reliability: The system's ability of breakdown free operation to the extent to which the system fails is reliability. It is calculated by the Mean Time between Failures. The verification of a system aims to notice defects and then take away them there by making it more dependable. A frequent change introduces the defects into the software affecting the reliability. Testing must be done to confirm that no defects have been introduced by the change after it has been implemented. The effect of an alteration on software reliability can be indirectly calculated by measuring its effect on the complication of the software and also can be calculated when the change is prepared.

5.1 Algorithm Used in our Methodology:

In the proposed methodology the genetic algorithm will be used. Genetic algorithm is computational model that is inspired from the biological inspiration.

Basic genetic algorithm:

1. **Start,** Generate the random population of n chromosomes i.e. to generate suitable solutions for the problem.
2. **New population,** Create a new population by repeating following steps until the new population is complete.

a. Selection**,** here we select the two parent chromosomes from a population according to their fitness.

b. Crossover, with a crossover probability crossover the parents to form a new offspring. If there is no crossover then offspring is an exact copy of parents

c. Mutation, with a mutation probability mutate new offspring at each position in chromosome.

d. Accepting, it means we have placed a new offspring in a new population.

3. **Replace,** use new generated population for a further run of algorithm.
4. **Test,** if the end condition is satisfied, then terminate and return the best solution in current population.
5. **Loop,** Go to step 2.

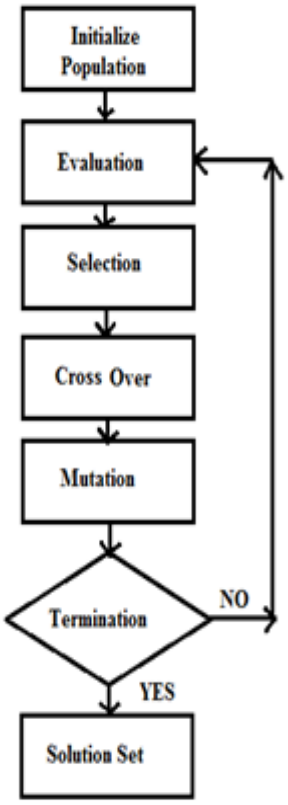

Figure 3.2: Flow Chart of Genetic Algorithm

5.2 Genetic Algorithm in Dynamic metric: Genetic algorithm works upon the basis of the knowledge of experts. The Genetic algorithm having the two unlike pools, in the pool 1 there are the evils which are present in the existing software, that are additional to the pool by the experts. In the pool 2, there are the solutions added corresponding to the evils introduced in pool 1. Now in the planned methodology both the pools will be used. As in this research work, the two problems from the pool 1 will be chosen, as there are many evils related to the dynamic metrics. These problems are related to the reliability and testability, then some factors in pool 2 will be introduced that will be helpful in solving the problem.

VI. CONCLUSIONS

This study indicates that Dynamic metrices gives better results results as compared to static metric because dynamic metrices gives conclusions about the dynamic dependencies between the various components of the software during run time of software. Dynamic metrices depends upon running code but static metrices are based on non executable code. Most of the work is done on dynamic metric which evaluate the attributes like inheritance, polymorphism, cohesion and coupling between the components of the software during the run time but least work is done on attributes like testability and reliability. When new features are added to the software then it may lead to new errors in the software that result in its poor quality. In our methodology we are adding two new factors named testability and reliability to the already existing dynamic metric using a genetic algorithm. Genetic algorithm is a computational model that find the best solution from the set of solution. By using the genetic algorithm we develop a new dynamic metric that has better performance as compared to already exiting dynamic and static metrices.

ACKNOWLEDGMENT

I would like to thanks CSE department of RIMT-IET Mandi Gobindgarh, Punjab.

REFERENCES

[1] Gregg Rothermel, Roland H.Untch, ChengyunChu,Mary Jean Harrold,(2001) "Prioritizing Test Cases for Regression Testing", CSE Journal Articles, Paper 9

[2] Y. Hassoun, R. Johnson, S. Counsell, (2004) "A Dynamic runtime coupling metric for meta-level architectures", Software Maintenance and Reengineering, pp. 339-346.

[3] Aine Mitchell, James F.Power, (2004) "Run-Time Cohesion Metrics: An Empirical Investigation

[4] Yua Jiang et.al, (2008) "Comparing design and code metrics for software quality prediction", 4th International workshop on Predictor models in software engineering, pp. 11-18.

[5] ParvinderS.Sandhu, Satish Kumar Dhiman, Anmol Goyal, (2009) "A Genetic Algorithm based Classification Approach for finding Fault Prone Classes", World Academy of Science Engineering and Technology.

[6] PayalKhurana, Puneet Jai Kaur, (2009) "Dynamic Metrics at Design Level", International Journal of Information Technology and Knowledge Management, Volume 2, No. 2, pp. 449-454.

[7] Er.Iqbaldeep Kaur, Dr. P.K.Suri, Er.AmitVerma, (2010) "Characterization and Architecture of Component Based Models", International Journal ofAdvanced Computer Science and Applications, Volume 1, No.6.

[8] Varun Gupta, Jitender Kumar Chhabra, (2011) "Dynamic Cohesion Measures for Object-Oriented Software", Journal of Systems Architecture, pp. 452–462.

[9] Deepak Arora, Pooja Khanna, AlpikaTripathi, Shipra Sharma, Sanchika Shukla (2011) "Software Quality Estimation through Object Oriented Design Metrics", International Journal 100 of Computer Science and Network Security, Volume 11, No.4.

[10] Dr. Gurdev Singh, ManikSharam, (2011) "Analysis of Static and Dynamic Metrics for Productivity and Time Complexity", International Journal of Computer Applications, Volume 30, No.1.

[11] Amjed Tahir, Stephen G.Mcdonell, (2012) "A Systematic Mapping Study on Dynamic Metrics and Software Quality", IEEE International Conference on Software Maintenance.

[12] Mehmet Kaya, James W. Fawcett, (2012) "A New Cohesion Metric and Restructuring Technique for Object Oriented Paradigm",IEEE 36th International Conference on Computer Software and Applications Workshops.