Inheritance of Dynamic Metrics for increase In software Performance

Dr.M.V.L.N.Raja Rao HOD,IT Dept
D.Hema Sindhu, IT,Dept
Ch.Sugunalatha IT Dept
P.Lakshmi Padmaja IT Dept

Gudlavalleru Engineering College. Gudlavalleru, India.

Abstract- *Software Quality is very essential from the perspective of "Programmer" as well as "Customers". For Programmers/Developers it should "Conformance to requirements" and for Customers it should have "Specification quality". Software Quality is measured by Software Metric. Software Metrics are of two types: Static Metric and Dynamic metrics. Static metrics does not evaluate the execution behavior of the software and gives less accurate results as system nature changes during actual execution of software but Dynamic Metric focus on collection of data during the actual execution of the software that provide more accurate results than Static metric. In the previous work most of the work is done in order to calculate the software attributes like cohesion, coupling, Inheritance, polymorphism through Dynamic Metric. In purposed work we are Inheriting the existing Dynamic metrics by adding two new factors namely " Testability" and "Reliability" and there by increasing the performance of software that results in the good quality of the software.*

Keywords: Static metrics Dynamic metrics, cohesion, coupling, reliability, testability.

I.INTRODUCTION

Nowadays, software products rapidly increased and its usage is not limited to specific people or corporation. It is now used in the most of human life activities. Therefore, the software product quality is increasingly being important and the users demanding higher quality than ever before. Software Quality is measured by Software Metric .A software metric is a standard of measure of a degree to which a software system or process possesses some property.

Software Engineering: is a systematic approach to the development, operation, maintenance and retirement of software. The nature of software has changed a lot over the years. The earlier applications used to run on single processor is the fixed output. But today with the drastic improvement in the technology applications are having the complex user interface and these applications run on the various systems concurrently like applications which support client server architecture. To estimate the performance of the application we need to define some set of rules. As a result we embrace the concept, strategies and practices of the software engineering. With the use of these concepts and strategies we can estimate the applications performance and other factors.

1.1 Categories of Software

A. Application Software is the general designation of computer programs for performing tasks. It consist of standalone Programs that solve a specific business need. Application software may be general purpose or have a specific purpose. Application software is used to control business functions in real time.

B. **System Software** is computer software designed to provide services to other software. It is a collection of one or more programs used to control and coordinate the hardware and other application software. Generally the system software may perform the following operations: Communicates with hardware components. Controls and monitors the proper use of various hardware resources like CPU, memory, peripheral devices like monitor, printer etc. Examples of system software include *operating systems*, computational science software, game engines, industrial automation, and software as a service applications.

C. Malicious Software, commonly known as malware is a computer software developed to harm the computer systems. Malware can be in the form of worms, viruses, trojans, spyware, adware and rootkits, etc.

II.SOFTWARE QUALITY

Software Quality is the The degree to which a system, component, or process meets customer or user needs or expectations. It follows two types of criteria. Internal Criteria, that is not visible to the user and it is code-dependent and is for developer only. External Criteria, which is an experience in operational mode by users when running the software**.**

2 .1 Need to measure software Quality

There are number of reasons which inspire us to measure software. Some of the reasons are as follow.

- To Increase the Productivity of Software.
- To advance the quality of software and to improve the reliability of the system.
- Minimize the future Maintenance requirements

- To advance the development process so as to increase product quality.
- In order to meet or satisfy the customer needs and expectations.
- We need measurement on software so that we can recognize and agree on product quality.
- make more accurate to estimate of Project cost, schedule, complexity and effort needed for software development process.

2.2 Software Quality as a Layered Technology

Software development is completely a layered technology. which is, to develop software one will have to go from one layer to another. The layers are related and each layer demands the fulfillment of the previous layer. For the software growth it has been divided into four layers. These layers are represented in the below figure 1.1.. If all the layers are considered during build process, it leads to maximum fulfillment of user requirements.

Fig 1.1:Layered Technology

1.Tools: Softwareengineering *tools* provide automated or semi-automated support for the process and the methods. When tools are combined so that information created by one tool can be used by another.

2. Methods: Software engineering *methods* provide the technical how-to's for building software. Methods will include all the activities like requirements analysis, design, program construction, testing, and support.

3. **Process: al**lows the development of software on time. It defines an outline for a set of key process areas (KPA's) that must be acclaimed for effective delivery of software engineering technology.

4.**Quality focus:** The bedrock that supports software engineering is a quality focus. The quality management is backbone of software layered technology which consists of Total Quality Management Tools, Six sigma methods etc.The software Product must fulfill the Customer, developer, user quality requirements

III.SOFTWARE QUALITY METRIC

3.1 Static Metrics

There are various software metrics have been used to measure the different features of software to improve the software quality. Static metrics are those which work upon the code which is not under execution, that is, non-executable code. In previous years, various techniques have been developed which works to get the exexution behavior of a program. These techniques are like program slicing, run time languages, instruction counts etc.Some Static metric used earlier are:

1. Lines of code(LOC), measures the size of the program by counting the number of instruction lines in the source code. By using LOC we can predicts the effort that is required to develop the software program.

2.Cyclomatic complexity, is a software metric (measurement), used to indicate the **complexity** of a program It is a measure of the number of linearly independent paths in a program's source code by representing the control flow graph of a program. .

3. Function point A function point is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points are used to compute a functional size measurement (FSM) of software. For estimation the user necessities are categorized into the forms input, output, inquiries, internal files and external interfaces.

4.Halstead Complexity tends to recognize the measurable properties of software and also interrelationship between them. The identified properties are the program length, program vocabulary, volume, difficulty and effort.

3.2 Dynamic metric:

Dynamic metrics are those which perform examination on the executable code or in other words running code. Dynamic metrics helps to calculate the dynamic behavior of an application at run-time. The outcome of dynamic measures are much more accurate than the static measures. Dynamic metrics includes the reliability modeling along with complexity measures. These metrics depends on the input given to the system so as to make the system run. In order to know that how many tests have failed, this can only be done dynamically, that is, when the program is running. The complexity of the dynamic behavior of real-time applications motivates a shift from static metrics to dynamic metrics: Dynamic metrics are separated according to following attributes:

1. Cohesion: It refers to how closely and strongly the modules relates to each other. It is expressed as high cohesion and low cohesion. The modules that have high cohesion are preferred more over the modules

with low cohesion.

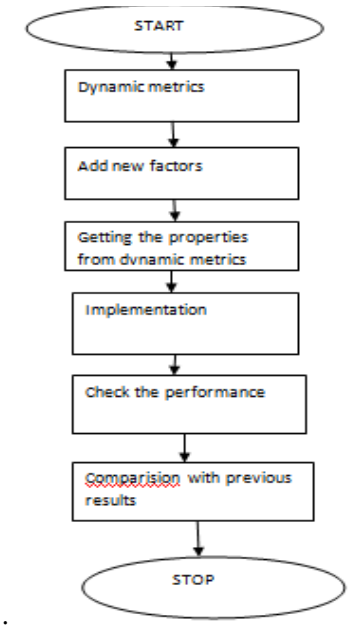
2. Coupling: It shows the dependency of one program module on another module. It is expressed in the form of low coupling and high coupling. Low coupling is correlated to high cohesion and vice-versa.

3. Inheritance: Inheritance measurement measures various aspects of inheritance such as deepness and wideness in a hierarchy and overriding complexity .The inheritance measurement gives us in sequence about the inheritance tree of the system. Inheritance is a key feature of the Object Oriented paradigm. This mechanism supports the class hierarchy design and captures the IS-A relationship between a super class and its subclass.

4. Polymorphism: Polymorphism is performing an additional functionality than once. Polymorphism is a software attribute that required to be measured vigorously. This metric was examined mainly in the context of software reusability next to with the determining total quality of the system. Various measurements are used for measuring polymorphism, which provides principle and accurate mechanism to detect and quantify the dynamic polymorphism.

IV. COMPARISON BETWEEN STATIC AND DYNAMIC METRICS

Static metrics does not assess the dynamic behavior at run time Dynamic measurement reflect the dynamic dependencies between the software components at run time



.

2. This measurement provides less accurate results than dynamic behavior will change at its run time.

Dynamic measurement provides more accurate results are collected during the actual execution of the software takes place.

3. Static measurement results are based on program structure and documents.

Dynamic measurement gives results according to the different types of inputs given to the system during run time.

4. Static measurement does not reflect key method like performance.

Performance is directly calculation by running the system using dynamic metric.

Proposed work:

The present work is about tremendously increasing the performance of the dynamic measurement by adding the factors. These factors help to increase the functionality of the software system. Now days, software quality measurement is used. These measurements are used to detect the quality of the various software products. The quality is nothing but excellence 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.TESTABILITY AND RELIABILITY

Testability: It is required in the Dynamic metrics. It is the non-functional requirement. It defines the property of measuring the ease of testing. It essentially measures the piece of code and functionality of the system. Testability allows the component to be tested in remoteness. A testable product is used for the entire implementation of the test scripts. When the testability takes place in the system, the customer reports the smallest number of defects. The testable products are easy and the cost to maintain product is also less. Testability is also significant for the maintainability of software product.

If we want to test a software product first we can check the part of a code to be tested. If the errors were encountered in part of that code, after that only the whole software will be checked. After checking the software we can increase the maintainability of the system. There are so many ways to show the testability requirements. In the testable systems, whenever user accepts the correct output, but the internal processes are not the similar as specify in the requirements, the system originate the defect

Reliability: The system's capacity of breakdown free operation to the extent to which the system fail is reliability. It is measured by the Mean Time between Failures. The mean time between failures is a

measure to measure hardware product is. The verification of a system aims to notice defects and then remove them there by creation it more reliable. Regular changes introduce the defects into the software affecting the reliability. Testing must be there to evaluate to check that no defects have been introduced by the vary after it has been implemented. The effect of an amendment on software reliability can be ultimately measured by measuring its effect on the involvedness of the software and also can be measured when the alter was prepared.

VI.GENETIC ALGORITHM

Genetic algorithm is computational model that is inspired from the biological inspiration.

Basic genetic algorithm:

1.Start: Create the random population of n chromosomes i.e. to generate suitable solutions for the problem.

2.New population: Generate a new population by repeating following steps awaiting the new population is complete.

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

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

c. transformation, with a mutation probability mutates new offspring at each position in chromosome.

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

1. **Replace:** use new generated population for a further run of algorithm.
2. **Test:** If the end condition is contented, then finish and return the best solution in current population.
3. **Loop:** Go to step 2.

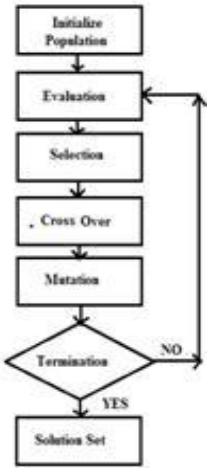


Figure 3.2: Flow Chart of Genetic Algorithm

VII.CONCLUSION

The results of this study indicate that there is growing interest in dynamic metrics in the software Engineering research area In respect to the many dimensions of software quality, it is clear that complexity- and maintainability-oriented dynamic metrics have been the most widely discussed in the literature; however the same high level of attention has not been directed to metrics for other quality dimensions, such as testability and reliability. It is concluded from the above discussion that dynamic metrics could be well suited to measure, and predict, testability. Measuring testability dynamically could be effective, particularly when considering different levels of testing (e.g., unit, integration, system) and the relationships between components. We conclude that most research now a date has addressed complexity- and maintainability-related aspects of dynamic metrics. A great deal of measurement focus has been given to factors such as coupling, cohesion and very little attention has been directed to other quality-related factors such as testability and reliability. In particular, we plan to investigate the feasibility of using dynamic metrics to measure other software quality characteristics such as reliability and testability.

ACKNOWLEDGEMENT

I would like to thanks IT department of GEC Gudlavalleru .

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.

[13] Mitsuhiro Nakamura, Tomoki Hamagami, (2012)"A Software Quality Evaluation Method Using The Change Of Source Code Metrics", IEEE 23rd International Symposium on Software Reliability Engineering Workshops.

[14] P.B. Nirpal, K.V. Kale, (2012) "Genetic Algorithm Based Software Testing Specifically Structural Testing For Software Reliability Enhancement", International Journal of Computational Intelligence Techniques, Volume 3, Issue 1, pp. 60-64.

[15] C.R. Kothari, Research Methodology: Methods and Techniques, New Age International Publishers, Rajasthan

[16] Antonia Bertolino, (2013) "An Orchestrated Survey on Automated Software Test Case Generation".

[17] Chayanika Sharma, SangeetaSabharwal, RituSibal, (2013) "A Survey on Software Testing Techniques using Genetic Algorithm", International Journal of Computer Science Issues, Volume 10, Issue 1.