Volume 4, Issue 10, October 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Survey paper on Quality of Service Analysis of Software Reliability and Testability

Vaid Singh
M.Tech Student in RIMT
India

Charnpreet Kaur
Assistant Professor in RIMT
India

Abstract: Quality of Software is very essential from the point of observation of "Developers" as well as "Consumers". For Developers it should "Conformance to requirements" and for Consumers it should have "Specification quality". Quality of Software is measured by Software Metric. Software Metric is the measurement of some property of Software. Software Metric are of two types: Static Metric and Dynamic metric. Static metric does not evaluate the changing behavior of the software and gives less precise results as system behavior changes during actual execution of software but Dynamic Metric focus on data collection 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 calculating software attributes like cohesion, coupling, Inheritance, polymorphism via Dynamic Metric. In our purposed word we are adding two new factors S named" Testability" and "Reliability" to the existing Dynamic metrics and hence increasing the performance of software that results in the good quality of the software.

Keywords: Dynamic metrics, reliability, testability, static metric, cohesion, coupling

I. INTRODUCTION

1.1 Software Engineering: The structure and the working of the software systems have passed through major changes in the last 30 years. The earlier applications used to run on single processor is the fixed output. But today with the improvement in the technology applications are having the complex user interface and these applications run on the various systems simultaneously 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 adopt the concept, strategies and practices of the software engineering. With the use of the software engineering concepts and strategies we can estimate the applications performance and other factors.

1.2 Categories of Software

A. Application Software the application software makes the use of computers to do some functional work as some entertainment functions separately from the fundamental functionality of computer.

B. System Software, are developed to function the hardware components and for fundamental functionality of the system and act as a platform for running application software. It comprises of, an operating system, which is a group of computer programs which are accountable to manage the computer resources and also provides some common services for the other software. The core parts of any operating system are supervisory programs, boot loaders shells and window system. An operating system includes application software to provide help to the user to do some helpful work on the system. The other part of system software is device driver, which is a program to operate and control any external device that is connected to a computer. The external devices that are associated externally to a computer system require a particular device driver. Utilities, is software made to facilitate the users in preservation of their computers.

C. Malicious Software, also termed as malware is a computer software developed to damage the computer systems. It is quite unwanted Malware is correlated with computer crimes.

II. SOFTWARE QUALITY

Software Quality is the measure to which a process of technique and mechanism of a system meets the necessities that are by now specified. It can also meet customers or users necessities. 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 a software. Some of the reasons are as follow.

- To inspect the cost and effort.
- To advance the production of software.
- To advance the quality of software and to raise reliability of the system.
- Decrease the future Maintenance requirements

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

2.2 Software Quality as a Layered Technology

A layered technology helps for constant improvement in software growth process. For the software growth it has been divided into four layers and each layer correlates to each other. These layers are represented in the figure 1.1. It displays all the four layers upon which the software engineering relies so as to develop superior software. If all the layers are considered during build process, it tends to fulfill maximum user requirements and helps to move smoothly to whole the engineering process.

Figure 1.1: A Layered Technology

1. **Tools:** These are the software engineering tools which provide sustain up for the other two layers, that is, process and methods. The tools are either programmed or semi-programmed.
2. **Methods:** The methods regularly state how to develop software. They contain a variety of phases of software development life cycle and offer special modeling techniques and the descriptive terminologies.
3. **Process:** It specifies the key process areas where the software is developed, organization of the deadlines, conforming the superiority of the software and focusing on maintenance.
4. **Quality focus:** Determines how well the software meets the functional and the non-functional requirements and helps to develop the more improved development approaches.

III. SOFTWARE QUALITY METRIC

3.1 Static Metrics

In recent years, various software metrics have been used to measure the different aspects of software as complexity to improve the software quality. Static metrics are those which work upon the code which is not running, that is, non-executable code. In previous years, various 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....Some Static metric used earlier are:

1. Source lines of code, measures the size of the program by counting the number of lines in the source code. It also predicts the effort that is required to develop the software program.

2. Cyclomatic complexity, It is used to compute the complexity of a program by representing the control flow graph of a program. It then measures the independent path to calculate the complexity.

3. Function point analysis is used to approximation the cost of the system with the help of previous projects. For estimation the user necessities are categorized into the forms inputs, outputs, inquiries, internal files and external interfaces.

4. Halstead Complexity tends to identify the measurable properties of software and also the relationship 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 investigation on the executable or in other words running code Dynamic metrics helps to calculate the dynamic behavior of an application at run-time. The results of dynamic measures are much more precise than the static measures. Dynamic metrics includes the reliability modeling along with complexity measures. These metrics depends upon 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 metrics measure various aspects of inheritance such as depth and breadth in a hierarchy and overriding complexity .The inheritance metrics give us information about the inheritance tree of the system. Inheritance is a key feature of the OO paradigm. This mechanism supports the class hierarchy design and captures the IS-A relationship between a super class and its subclass

4. Polymorphism: Polymorphism means using the similar function additional than once. Polymorphism is a software characteristic that required to be measured 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 an objective and precise mechanism to detect and quantify the dynamic polymorphism.

IV. COMPARISON BETWEEN STATIC AND DYNAMIC METRIC

1. Static metrics does not evaluate the dynamic behavior at run time
Dynamic metric reflect the dynamic dependencies between the software components at run time
2. Static metric provides less accurate results as system behavior changes during run time
Dynamic metric provide more accurate results as results are collected during the actual execution of the software
3. Static metric results are based on program structure and documentation
Dynamic metrics gives results according to the different types of input given to the system during run time
4. Static metric does not reflect key attributes like performance
Performance is directly calculation by running the system using dynamic metric.

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 isolation. 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 amount 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.

When software is tested, firstly a piece of code is tested. The errors are found in that piece of code. After that the whole system is tested. Hence testability increases the maintainability of the system. In the testable systems, whenever user receives the correct output, but the internal processes are not the similar as specify in the requirements, the system originate the defects. There are many ways to show the testability requirements.

Reliability: The system's ability of failure free operation to the extent to which the system fails is reliability. It is measured by the Mean Time between Failures. The verification of a system aims to notice defects and then eliminate them there by making it extra reliable. Regular changes introduce the defects into the software affecting the reliability. Testing must be done to verify that no defects have been introduced by the change after it has been implemented. The effect of a amendment on software reliability can be ultimately measured by measuring its effect on the complexity of the software and also can be measured when the amend is prepared.

VI. GENETIC ALGORITHM

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 mutates 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.

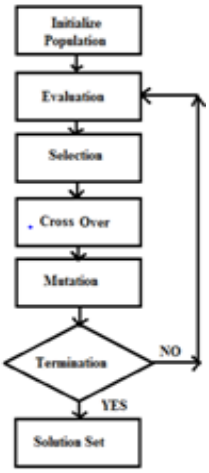

Figure 3.2: Flow Chart of Genetic Algorithm

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 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.

[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.

