INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

A REVIEW ON PERFORMANCE OF DYNAMIC QUALITY METRICS BY ADDING TESTABILITY AND RELIABILITY

Lovepreet Kaur[1], Rupinder Kaur[2]

[1]Student, Lovely Professional University,
Phagwara, Pin no. 144402
luvpreet_shah@yahoo.com

[2]Asst. Professor, Lovely Professional University
Phagwara, Pin no. 144402
rupinder.16835@lpu.co.in

Abstract: *Dynamic metrics is the part of object oriented metrics. In the dynamic metrics, the quality is very necessary, because it helps to enhance the system performance. Software Quality is the measure to which a process of method and components of a system meets the requirements that are already specified. A software metrics is the measurement of some properties of software. Software metrics can be classified into the three categories: Product metrics describe the characteristics of the product, these characteristics include size, complexity, design features, performance, and quality level. Process metrics can be used to improve software development and maintenance. Project metrics describe the project characteristics and execution. In our purposed work, we are going to enhance the system performance by adding the two factors: testability and reliability. It also helps to reduce the time.*

Keywords: *dynamic metrics, reliability, testability, design features, performance.*

1. INTRODUCTION

Software Engineering: The nature and complexity of the software systems had changed significantly in the last 30 years. The previous applications run on single processor and produce fixed output .But with the advancement in the technology application are having the complex user interface and these applications run on the various systems simultaneous like applications which support client server architecture. Today applications can run on various operating systems due to the nature and complexity of the applications we need to evaluate the performance and other factor of the application .To evaluate the performance of the application we need to define some set of rules. Therefore, we adopt the concept, strategies and practices of the software engineering. With the use of the software engineering concepts and strategies we can evaluate the applications performance and other factors. [1] We have to check the some major failures that will leads to software failure before delivering the application to the user.

The changing nature of software: We classify the software according to nature and field in which that software is used:

System software: Infrastructure soft-ware comes under this category like compilers, operating systems, editors, drivers, etc. Basically system software is a collection of programs to provide service to other programs

Real time software: This software is used to monitor, control and analyze real world events as they occur. An example may be software required for weather forecasting. Such software will gather and process the status of temperature, humidity and other environmental parameters to fore cast the weather.

Embedded software: This type of software is placed in “Read-Only-Memory (ROM)”of the product and controls the various functions of the product. The product could be an aircraft, automobile, security system, signaling system, control unit of power plants, etc. The embedded software handles hardware components and is also termed as intelligent software.

Business software: This is the largest application area. The software designed to process business applications is called business software. Business software could be payroll, file monitoring system, employee management, and account management. It may also be a data ware housing tool which helps us to take decisions based on available data. Management information system, enterprise resource planning (ERP) and such other software are popular examples of business software.

Personal computer software: The software used in personal computers is covered in this category. Examples are word processors, computer graphics, multimedia and animating tools, database management, computer games etc. This is a very upcoming area and many big organisations are concentrating their effort here due to large customer base.

Artificial intelligence software: Artificial Intelligence software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straight forward analysis [PRESOI]. Examples are expert systems, artificial neural network, signal processing software etc.

Software Quality: Software Quality is the measure to which a process of method and components of a system meets the requirements that are already specified. We can also says that in which it can meets customers or users requirements also . [2]It follows two types of criteria.

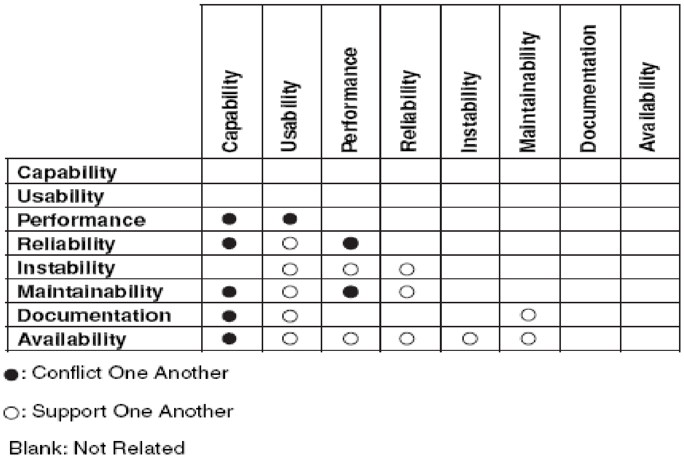
- Internal Criteria
- External Criteria

Internal Criteria: It is not visible to the user and it is code-dependent. It is for developer only.
External Criteria: It is an experience in operational mode by users when running the software**.**

Software Quality Metrics:
A software metrics is the measurement of some properties of software. Software metrics can be classified into the three categories:

- **Product metrics** describe the characteristics of the product; these characteristics include size, complexity, design features, performance, and quality level.
- **Process metrics** can be used to improve software development and maintenance.
- **Project metrics** describe the project characteristics and execution.

The software quality metric encapsulates the above two attributes, addressing the mean time to failure and defect density within the software components. Example of software quality metrics:



	Capability	Usability	Performance	Reliability	Instability	Maintainability	Documentation	Availability
Capability								
Usability								
Performance	●	●						
Reliability	●	○	●					
Instability		○	○	○				
Maintainability	●	○	●	○				
Documentation	●	○				○		
Availability	●	○	○	○	○	○		

●: Conflict One Another

○: Support One Another

Blank: Not Related

Figure1: Software quality metrics

It shows the various factors of the software metrics.

2. LITERATURE SURVEY

Antonia Bertolino, An Orchestrated Survey on Automated Software Test Case Generation, (2013) : It has introduced about software testing and its goal and scope. Further explain about the breakdown rationale of the software testing and different types of terminology used in software testing. [3] In this paper decomposition of testing and definitions are mentioned.

Sanjay Misra, Ibrahim Akman and Murat Koyuncu, An inheritance complexity metric for object-oriented code, a cognitive approach, (2011): In this paper author discuss about the use of software metrics. Software metrics can be used to improve the quality and productivity of the system. [4] For designing the object oriented code complexity factor can be used. In the object oriented system inheritance is used to check the complexity of the system. The design of the system is difficult and expensive to change. The object oriented software development is expensive due to some features like encapsulation, inheritance, polymorphism and reusability. The software development is affected by the complexity of the system. This complexity create problem in the understandability of the software design. The complexity of the metrics can be calculated by the traditional metrics. The object oriented metrics do not consider the cognitive characteristics in calculating the code complexity. The cognitive complexity is defined as the mental burden on the user who deals with the code. The high cognitive complexity is undesirable due to many reasons, as, fault tolerance and reduced maintainability. Cognitive designs give the information about the design of object oriented systems. High cognitive complexity leads to the poor design. It is difficult to manage.

Er.Iqbaldeep kaur et.al, Characterization and Architecture of Component Based Models,(2010): Emphasized that component analysis based on the reliability of the software typically takes the reliability of the component as a component invariant property in itself does not account for the fact that changes when the context changes interacting component due to a different profile of system operation .[5] This article presents an analysis approach to characterize sub domain path component reliability model based on the architecture, and provides improved composition algorithms to solve the model. The analysis of the implementation of a case study indicates that our model is able to fully capture the effect of different operational profiles on the overall reliability of the system, from the aspects of the transition probability and component reliability. This ability will benefit reliability analysis of more accuracy and flexibility.

Deepak Arora, Pooja Khanna and Alpika Tripathi, Software Quality Estimation through Object Oriented Design Metrics, [2011]: Software metrics are used for measure quality. It measures the quality in terms of software performance and reliability. Software metrics are use to measure progress of code during development. It has the relationship with cost and time that used in the development of the software design. These metrics are focus on the issues like complexity, reliability and robustness of the software system. Object oriented metrics provides parameters through which complexities and quality related issues of any software can be estimated. In this paper author discuss about the three metrics, as MOOD Metrics, CK Metrics, and QMOOD Metrics. [6] Software Metrics defined as the quality of a software objects. In the software metrics quality maps are used for adding new parameters like Extensibility, Reusability, efforts, manageability and cost.

Varun Gupta, Jitender Kumar Chhabra, (2010), Dynamic cohesion measures for object-oriented software: In this paper, author describes the object oriented cohesion metrics. The measurement of object level dynamic cohesion gives better insight into the behavioral aspects of the system. In this paper, dynamic cohesion metrics are introduced. The dynamic cohesion metric provide scope of cohesion measurement up to object level. [7] It used object oriented features during the measurement, these features are inheritance, polymorphism and dynamic binding. A dynamic analyzers tool is developed using aspect oriented programming to perform dynamic analysis of java applications for the purpose of collecting run time data for computation of the proposed dynamic cohesion measures.

3. PURPOSED 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. In the previous days, the quality of the software product is depends upon the features add. Suppose a product has the high features than the other one then it becomes the best quality product. But there are many other problems has 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.

4. METHODOLOGY

In our purposed methodology we design a new metrics by adding some factors in the existing metrics**.** here we add the two factors in the dynamic metrics. These factors are help to enhance the performance of dynamic metrics. These factors are:

- **Testability**:

Testability is required in the Dynamic metrics. It is the non functional requirement. It defines the property of measuring the ease of testing. It basically measure the piece of code and functionality of the system. Testability allows the component to be tested in isolation. Testability is necessary in many ways. A software development lifecycle consist the requirements gathering, analysis, design, coding, testing, implementation, and maintenance. A testable product is used for the complete execution of the test scripts. When the testability is take place in the system, the customers reports the minimum number of defects. The testable products are easy and the cost to maintain product also less. Testability is an important aspect for the maintainability of software product. When software is tested, firstly a piece of code is tested. The errors are finds out 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 same as specify in the requirements, the system found the defects. There are many ways to show the testability requirements. [8]

- **Reliability:**

It is most commonly measured by the Mean Time between Failures. Changes that reduce reliability should be avoided. The software verification process aims to detect and remove defects and make it more reliable. Changes that introduce defects into the software make it less reliable. Tests should be used to verify that no defects have been introduced by the change after it has been implemented. Reliability can be reduced by reusing components that have not been developed to the same standards as the rest of the software. The effect of a modification on software reliability can be estimated indirectly by measuring its effect on the complexity of the software. This effect can be measured when the change is designed.

Algorithm Used in our methodology:

In our purposed methodology we use the genetic algorithm. Genetic algorithm is computational model that is inspired from the biological inspiration.

Basic genetic algorithm:

(Start) generate the random population of n chromosomes i.e. generate suitable solutions for the problem

(new population) Create a new population by repeating following steps until the new population is complete.

(selection) select two parent chromosomes from a population according to their fitness.

(crossover) with a crossover probability cross over the parents to from a new offspring. If no crossover was performed, offspring is an exact copy of parents

(mutation) With a mutation probability mutate new offspring at each position in chromosome.

(accepting) place a new offspring in a new population.

(replace) use new generated population for a further run of algorithm

(test) if the end condition is satisfied,stop,and return the best solution in current population.

(loop) Go to step 2.

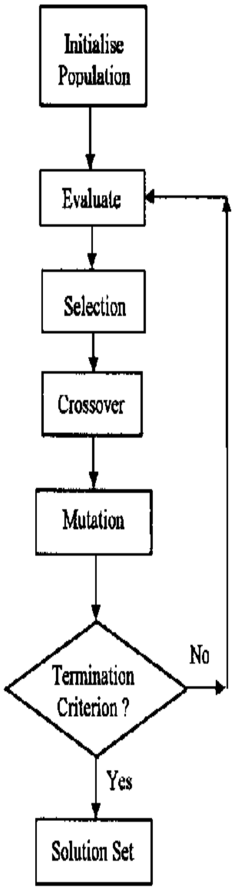


Figure 2: Flow chart of genetic algorithm

Genetic Algorithm in Software Engineering: genetic algorithm is worked on the basis of the experience of experts. The Genetic algorithm having the two different pools, In the pool 1 there are lots of problem present, that added by the experts. In the pool 2 there are several solutions added corresponding to that problem. Now we use both the pools in our proposed methodology. As in our case, we choose the two problems from the pool 1, there are many problems related to the dynamic metrics. As in our case the problem is related to the reliability and testability, so we add the some factor in pool 2, that are help to solve the problem. As:

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

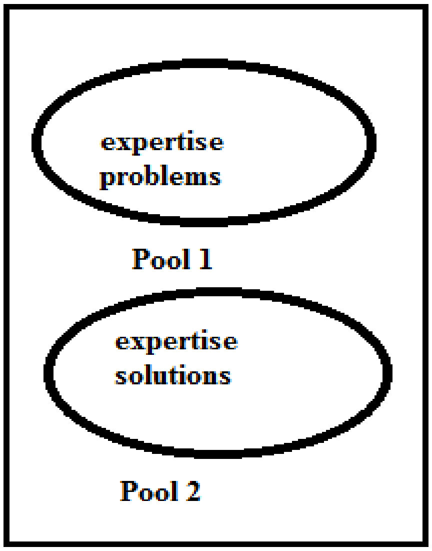


Figure 3: Expert's pool

In the fig 3.2, the expert pool is shown, here there are two pools. In the first pool there is problems that are defined by the experts whereas in the second pool the solutions are shown that are given by the expertise.

REFERENCES

[1].http://www.augustana.ab.ca/~mohrj/courses/2003.fall/csc220/lecture_notes/SE_definition.html
[2] book: metrics and models in software quality enigeneering by Stephen h.kan
[3] Antonia Bertolino, J. Jenny Li, Hong Zhu, An Orchestrated Survey on Automated Software Test Case Generation, the Proceedings of IEEE/ACM Workshops on Automation of Software Test (AST'06 – AST'12). URL for AST'13: http: //tech.brookes.ac.uk/AST2013/ Preprint submitted to Journal of Systems and Software February 11, 2013
[4] SANJAY MISRA, IBRAHIM AKMAN and MURAT KOYUNCU, An inheritance complexity metric for object-oriented code: A cognitive approach, Sa¯dhana¯ Vol. 36, Part 3, June 2011, pp. 317–337._c Indian Academy of Sciences
[5] Er.Iqbaldeep kaur, Dr.P.K.Suri, Er.Amit Verma, Characterization and Architecture of Component Based Models, IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 1, No. 6, December 2010
[6] Deepak Arora, Pooja Khanna and Alpika Tripathi, Shipra Sharma and Sanchika Shukla, Software Quality Estimation through Object Oriented Design Metrics, IJCSNS International Journal 100 of Computer Science and Network Security, VOL.11 No.4, April 2011
[7] Varun Gupta , Jitender Kumar Chhabra, Dynamic cohesion measures for object-oriented software, Journal of Systems Architecture 57 (2011) 452–462
[8] Payal Khurana & Puneet Jai Kaur ,DYNAMIC METRICS AT DESIGN LEVEL, International Journal of Information Technology and Knowledge Management July-December 2009, Volume 2, No. 2, pp. 449-454