MUTATION TESTING FOR C-SHARP PROGRAMS

Sujata Pawar
Student: Dept. of Computer Engg, D. Y. Patil College of Engg, Akurdi, Pune, India

Mrs. V. L. Kolhe
Guide: Dept. of Computer Engg, D. Y. Patil College of Engg, Akurdi, Pune, India

ABSTRACT

Program testing is the important phase of program development process. This phase can be easily missed by program developers because of their limited time to complete the project. Since, program developers finish their programs nearer to the delivery time; they don’t get enough time to test their program by creating effective test cases. One of the major difficulties in Programs program is the generation of test cases that satisfy the given adequacy criterion also with, creating manual test cases is a tedious work for program developers in the final rush hours. Testing is a critical activity of program design. Exhaustive testing of a program is not possible so different techniques are invoked. Mutation testing is a powerful testing technique for generating program tests and evaluating the quality of program. We have to approach different mutation for OOP adoption. In this paper first we explain the different types of mutation operator for these operators include the OOP features of fault detection are used are based on specific oriented fault. Mutation test trials, and high reliability to improve quality assurance programs of great potential for program units to test In this paper a technique that a mutation testing test cases test case generation technology program execution trace, after coding test cases can be generated to produce has been proposed.

Keywords: Mutation Testing, Programs Testing, Object-Oriented Concepts.

I. INTRODUCTION

Programs testing are an important phase of Programs development life cycle. The entire test is usually a problematic test strategies that work, a sufficiently effective test cases to find out to be revealing mistakes are faced with how to build a minimal set? That's why the mutation testing strategy we are testing different techniques and different testing techniques to object-oriented
programming to a fault-based testing. in particular, the quality check can reveal the type of fault and test sets test input data used in the test by testing measures.

Mutation generates simple syntactic deviations test methods and mutants representing a specific programming errors from the original program mutation changes the ways such as an arithmetic operator says replace a + *, /, which represents an incorrect operator using the programmer intended. If we say that killing a mutant test cases in term of the original mutant program program output to distinguish otherwise mutants alive. "there In the case of the two possibilities can be

**Live mutant:** - Either the mutants that we launched is a mutant and produce equivalent results and it can't be killed. in the second case insufficient means our test case by adding new test cases so we can improve the original symbols and non-equivalent mutant is not able to distinguish between our test set.

Mutation testing is a fault test set based on whether the test measures the quality of trial tactics and tested by test used to test input data can reveal certain types of guilt. the quality of the test data test examiners mutation to interact with them helps you to create test data. due to faulty program test cases for each incorrect output (failed) with the goal of producing these defective program to execute Is therefore used the word mutation; And mutants defective program to kill a mutant originally when he fails when it happens, is represented by the mutant mutant mistakes has been found, because the test process to be dead and no longer needed.

The rest of the paper is organized as follows. Related work is reviewed in Section II. In section III we are presenting proposed method and architecture. In Section IV, presents the illustrative simulation results and discussions. Section V concludes the paper.

**II. LITERATURE SURVEY**

If testers want to test functional requirements, black box testing techniques can use black box testing does not need knowledge of how c-sharp, the program is programmed to test prediction program are defined by the specifications. Inject test data program testers to execute, specified test compares actual results with oracle. By contrast White box testing need knowledge of how c-sharp is programmed. White-test box, testing oracle path or statement which has been performed in the past. These coverage criteria. coverage are three main types of criteria: statement, coverage, branch coverage, and path coverage. Statement coverage reports whether each statement is encountered by the test suite or not. Branch coverage reports whether every branch structure has been executed for true and false condition in each branch. Finally, path coverage reports whether all possible paths in function has been tested.

Object-oriented languages, there is more than one complex software structure is structural. traditional test approaches may not be enough for testing. the combination of those two traditional approaches gray box testing. gray box testing, generates the expected test data structure and specifies the behavior of the high level system design based on. grey box testing coverage criteria check the white box method and all possible coverage path. In addition, in the case of black box testing testing criteria arising as functional requirement should be satisfied with.

**III. PROPOSED ARCHITECTURE**

There are two kinds of mutation operators available namely statement level operators, method level operators and class level operators.

Statement level mutation operators program being tested involves building a set of mutant of programs. Each program is a different mutation of mutant origins and syntactic changes that a single
Mutation is a program used to statements, other operand or operands instead of a single operand with the constant substitution operators (ORO).

Expression Modification Operators (EMO) – Replacing an operator or inserting a new operator.

As we have discussed in previous section first four types of operator are common to object-oriented programming. The remaining two Java-specific features and Common Programming Mistake is C-sharp specific. Below is a given list for different types of operators

3.1. Access Control (Information Hiding)

Operator:- AMC (access modifier change) access level is a common source of mistakes. Poor access to definitions not due to fault at the beginning but when integrated with other classes can lead to faulty practices. C-sharp a reach as four levels: public, private, protected and if left unspecified for AMC operator package variables and methods changes the access level for the operator with other options Fixed Java access mode changed. AMC built by three mutants as the operator for example, access to a private variable mode, the role of the operator AMC examiners accessibility testing to generate enough test cases to guide, ensure that field should distinguish this variable test set when its access modifier public/protected/private default mode is exactly to order.

The original Code
private int a;
AMC Mutants:-
public int a;
protected int a;int a;

3.2 Inheritance Feature

Incorrect use of inheritance can leads to a number of faults Operators. Operator Name Descriptions
IHD: - Hiding variable deletion
IHI: - Hiding variable insertion
IOD: - Overriding method deletion
IOP: - Overriding method calling position change
IOR: - Overriding method rename

IV. PRACTICAL RESULTS

This proposed framework the practical implementation to c# language used to test cases we have used these programs c# program General initialization, the main structural programme, and with other methods.

Following figure 1 is showing the main framework of mutation testing of C# program.
Following figure 2 is showing the match code of two programs. Here we check whether input code & test file code matches or not, if matches then which pattern is match.

![Figure 2: Match code of file](image)

From the above figure its showing that the current proposed method resulted into more accurate with use of heterogeneous test cases as compared without using the same.

Not only the accuracy, but also the speed of proposed approach is improved as compared to existing one. This is showing in below figure 3.

![Figure 3: Testing with different mutants](image)

V. CONCLUSION

This paper uses the object-oriented features of mutation operators for test presents a comprehensive set of mutation operators these OO defect and that gives them a firm theoretical basis are based on an extensive list of. as a result, they fix many problems these mutation operators to support Java interclass level testing with an emphasis on the integration of aspects of drafting. And testers access control, inheritance, polymorphism and overloading as to find faults with the use of language features. Thus, this provides a way to improve the reliability of OO software.
VI. REFERENCES

[12] MuJava : An Automated Class Mutation System Yu- Seung Ma, Jeff Offutt and Yong Rae Kwon
[13] ise.gmu.edu