Mutation Testing in Software Testing: Mutant Score & Analysis Example

Mutation Testing

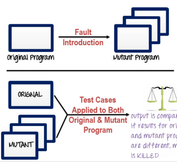
Mutation Testing is a type of software testing in which certain statements of the source code are changed/mutated to check if the test cases are able to find errors in source code. The goal of Mutation Testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code.

The changes made in the mutant program should be kept extremely small that it does not affect the overall objective of the program. Mutation Testing is also called Fault-based testing strategy as it involves creating a fault in the program and it is a type of White Box Testing which is mainly used for Unit Testing.

Mutation was originally proposed in 1971 but lost fervor due to the high costs involved. Now, again it has picked steam and is widely used for languages such as Java and XML.

In this tutorial, you will learn-

How to execute Mutation Testing?



Following are the steps to execute mutation testing(mutation analysis):

Step 1: Faults are introduced into the source code of the program by creating many versions called mutants. Each mutant should contain a single fault, and the goal is to cause the mutant version to fail which demonstrates the effectiveness of the test cases.

Step 2: Test cases are applied to the original program and also to the mutant program. A Test Case should be adequate, and it is tweaked to detect faults in a program.

Step 3: Compare the results of an original and mutant program.

Step 4: If the original program and mutant programs generate the different output, then that the mutant is killed by the test case. Hence the test case is good enough to detect the change between the original and the mutant program.

Step 5: If the original program and mutant program generate the same output, Mutant is kept alive. In such cases, more effective test cases need to be created that kill all mutants.

How to Create Mutant Programs?

A mutation is nothing but a single syntactic change that is made to the program statement. Each mutant program should differ from the original program by one mutation.

Original Program	Mutant Program
If (x>y)	If(x<y)
Print "Hello"	**Print "Hello"**
Else	**Else**
Print "Hi"	**Print "Hi"**

What to change in a Mutant Program?

There are several techniques that could be used to generate mutant programs. Let's look at them

Operand replacement operators	Expression Modification Operators	Statement modification Operators
Replace the operand with another operand (x with y or y with x) or with the constant value.	Replace an operator or insertion of new operators in a program statement.	Programmatic statements are modified to create mutant programs.

Example-
 	
If(x>y) replace x and y values
 	
If(5>y) replace x by constant 5
	Example-
	
	If(x==y)
	
	We can replace == into >= and have mutant program as
	
	If(x>=y) and inserting ++ in the statement
	
	If(x==++y)
		Example-
		
		Delete the else part in an if-else statement
		
		Delete the entire if-else statement to check how a program behaves
		
		Some of sample mutation operators:
		
		• GOTO label replacement
		• Return statement replacement
		• Statement deletion
		• Unary operator insertion (Like – and ++)
		• Logical connector replacement
		• Comparable array name replacement
		• Removing of else part in the if-else statement
		• Adding or replacement of operators
		• Statement replacement by changing the data
		• Data Modification for the variables
		• Modification of data types in the program

Automation of Mutation Testing:

Mutation testing is extremely time consuming and complicated to execute manually. To speed up the process, it is advisable to go for automation tools. Automation tools reduce the cost of testing as well.

List of tools available -

- Stryker
- PIT Testing

Types of Mutation Testing

In Software Engineering, Mutation testing could be fundamentally categorized into 3 types- statement mutation, decision mutation, and value mutation.

1. **Statement Mutation** - developer cut and pastes a part of a code of which the outcome may be a removal of some lines
2. **Value Mutation**- values of primary parameters are modified
3. **Decision Mutation**- control statements are to be changed

Mutation Score:

The mutation score is defined as the percentage of killed mutants with the total number of mutants.

- Mutation Score = (Killed Mutants / Total number of Mutants) * 100

Test cases are mutation adequate if the score is 100%. Experimental results have shown that mutation testing is an effective approach for measuring the adequacy of the test cases. But, the main drawback is that the high cost of generating the mutants and executing each test case against that mutant program.

Advantages of Mutation Testing:

Following are the advantages of Mutation Testing:

- It is a powerful approach to attain high coverage of the source program.
- This testing is capable comprehensively testing the mutant program.
- Mutation testing brings a good level of error detection to the software developer.
- This method uncovers ambiguities in the source code and has the capacity to detect all the faults in the program.
- Customers are benefited from this testing by getting a most reliable and stable system.

Disadvantages of Mutation Testing:

On the other side, the following are the disadvantages of Mutant testing:

- Mutation testing is extremely costly and time-consuming since there are many mutant programs that need to be generated.
- Since its time consuming, it's fair to say that this testing cannot be done without an automation tool.
- Each mutation will have the same number of test cases than that of the original program. So, a large number of mutant programs may need to be tested against the original test suite.
- As this method involves source code changes, it is not at all applicable for Black Box Testing.

Conclusion:

Do you want exhaustive testing of your application? The answer is Mutation testing. It is the most comprehensive technique to test a program. This is the method which checks for the effectiveness and accuracy of a testing program to detect the faults or errors in the system.

Prev Report a Bug Next