**Mutation Testing**

**I. Overview**

Mutation testing is a method of software testing in which a program or a source code is deliberately manipulated, followed by a suite of tests against the mutated code. The mutations introduced to the source code are designed to imitate common programming errors. A good unit test suite typically detects the program mutations and fails automatically.

The changes in the mutant program are kept extremely small, so it does not affect the overall objective of the program. The goal of Mutation Testing is to assess the quality of the test cases which should be robust enough to fail the mutant code.

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

**II. Applying mutation testing**

Generally, the steps to execute mutation testing are:

1) Introducing faults 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.

2) Applying test cases 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.

3) Comparing the results of the original and the mutant program.

4) If the original program and the mutant program generate different outputs, 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.

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

The mutation score is calculated based on the formula:

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

The minimum score is 0, the maximum one is 100, with more being better.

**III. Types of mutation testing**

Mutation testing can be fundamentally categorized into 3 types – statement mutation, decision mutation and value mutation

1) Statement mutation: Changes done to the statements by deleting or duplicating the line, which might arise when a developer is copy pasting the code from somewhere else.

2) Decision mutation: The decisions/conditions are changed to check for the design errors. Typically, one changes the arithmetic operators to locate the defects and also we can consider mutating all relational operators and logical operators (AND, OR , NOT).

3) Value mutation: An attempt to change the values to detect errors in the programs. We usually change one value to a much larger or a much smaller value. The most common strategy is to change the constants.

**IV. Tools which automatically generate mutant programs in Java**

Jester (<http://jester.sourceforge.net/>)

PIT (<http://pitest.org/downloads/>)

µJava (<https://cs.gmu.edu/~offutt/mujava/#Links>)

Jumble (<https://sourceforge.net/projects/jumble/files/>)