Mutation Testing

Mutation testing is a type of software testing where the source code is deliberately altered (mutated) to evaluate the effectiveness of test cases. The goal is to check whether existing test cases can detect these changes, ensuring the test suite is robust.

How Mutation Testing Works

Mutation testing involves the following steps:

1. Mutant Creation: Small modifications (mutants) are introduced in the source code. These modifications can be:
   - Changing arithmetic operators (+ to −, * to /).
   - Replacing logical operators (&& to ||).
   - Modifying conditional statements (< to <=).
   - Removing statements (deleting a line of code).
2. Test Execution: The existing test suite is executed against these modified versions of the program.
3. Mutation Analysis:
   - Killed Mutants: If the test case fails after a mutation, the mutant is "killed," meaning the test detected the change.
   - Survived Mutants: If the test passes despite the change, the mutant "survives," indicating a weakness in the test suite.
4. Test Suite Improvement: If too many mutants survive, the test cases need improvement to cover those untested scenarios.

Example:
<u>Original Code:</u>

```
public int add(int a, int b) {
   return a + b;
}
```

<u>Mutant Code (After Mutation)</u>

```
public int add(int a, int b) {
   return a - b;
}
```

If the test suite is strong, the test case will fail because *add(2, 3)* will return -1 instead of 5. The mutant is killed.
If the test suite lacks coverage (e.g., no test cases for the *add* method), the mutant survives, indicating a flaw in the test cases.

Mutation Score

$$MutationScore = (\frac{KilledMutants}{TotalMutants}) * 100$$

High Score (close to 100%) → Good test coverage.
Low Score → Weak test cases, requiring improvement.

How to conduct mutation testing

1. Write a unit test.
2. Write code to go through the test.
3. Run the mutation test with the code, ensuring the code and test work.

4. Make some mutations to the code and run the new mutated code through the test. Every mutant should have one error to validate the test's efficiency.
5. Compare the results from the original code and the mutated version.
    1. If the results don't match, the test successfully identified and executed the mutant.
    2. If the test case produced the same result between the original and mutant code, the test failed and the mutant is saved.
6. A mutation score can also be calculated. The score is given as a percentage that's determined using the formula noted above.