Pit Mutation Testing

Code Coverage:

When we execute any test case it calls the source and based on that it calculates code coverage

Unit cases contained 3 steps

1. Initializing
2. Calling actual method
3. Comparison i.e., Checking expected result with actual result

Suppose the comparison step is missing, so whenever there is any change in source code it will not break/fail the test case. But it will remain cover the code coverage.

One more scenario

Suppose there is a condition if (value <10) and we wrote a test case with value=5. So, it is passed. But in future the source code got changed and the logic changed to if (value <9/8/7). Then also our test cases pass. So, when writing the test cases the boundary condition need to be taken care.

Here is the point to check the Quality. So, we have the quantity and no of test cases are there but what about quality.

Here mutation test case came to picture. In run time it changes some value like change the boundary condition, remove one line or modify some value and check our test cases whether it is failing or not.

The process of changing the source code is called as mutant. There is different type of mutants are present. Each mutant has two concepts i.e., Survive or Killed

Mutant will try to change the business logic and call our test case, if our test case is failing then mutation is killed that is it is a good quality test case. If the test case is pass then the mutant is survived i.e., the quality is bad

What is Mutation Testing

1. Faults (or mutations) are automatically seeded or inserted into your code, then your test is run. If your test fails then the mutation is killed, if your tests pass then the mutation lived.

2. The quality of your tests can be gauged from the percentage of mutations killed.

Why Mutation Testing

1. Traditional test coverage (i.e., line, statement, branch, etc.) measures only which code is executed by your tests.

2. It does not check that your tests are actually able to detect faults in the executed code. It is therefore only able to identify code that is not executed

3. The most extreme examples of the problem are tests with no assertions. Fortunately, these are uncommon in most code base

4. As it is actually able to detect whether each statement is meaningfully tested, mutation testing is gold standard against which all other types of coverage as measured

What is the Goal of Mutation code coverage?

1. Identify weakly tested pieces of code (those for which mutants are not killed)

2. Identify weak tests (that those never kill mutants

3. Compute the mutation score

4. Learn about error propagation and state infection in the program.

PIT – Java Library for Mutation Testing

PIT is Fast, Easy to use, actively developed, Actively Supported

The reports produced by PIT are in an easy-to-read format combining line coverage and mutation coverage