Automatic JUnit Generation using Concolic and Mutation Testing

Abstract

In the development of software, the testing process plays a crucial role which helps to avoid risks by identifying errors and improving the cost and adaptability of the software. It is important to ensure that the software should not result in any failures and may lead to being very expensive in the later stages of development if not tested properly. Unit testing is the method of verifying the smallest piece of testable code against its purpose. Improper testing or human-based errors in manual testing of the software lead to potential defects in production and also causes human loss due to the failures caused by the software glitch. Code coverage is the metric that helps to detect such code segments and tells about unit testing practices done by the developers to target all areas of the code at least once. This work is the development of a tool that is capable of identifying logical errors and achieves code coverage by creating the unit test case for every feasible and possible path. Problem solvers like S3 developed by Microsoft are used to evaluate the mathematical expressions in the code. The concept of Concolic Testing (Java Path Finder) and Mutation Testing (PIT) strategies are adapted to achieve code coverage and identify logical errors. In addition, this tool can generate the JUnit test suites automatically.

1 Description of the work

This tool benefits the programmers to get the syntactically correct JUnits upon a button click. This project is developed to achieve high code coverage, handle the run-time exceptions, catch the logical errors and unhandled exceptions, implement mocking, and catch the logical errors present in the code.
The tool does the following:

1. Generates the JUnit by covering all paths of the branching statements within the code where a test case is generated for every path to achieve the optimum number of test cases and maximum code coverage.
2. Verifies the significance of the test cases and their quality to catch logical errors.
3. Generates the JUnits for the classes implementing abstract classes and Interfaces as well.

2 Methods and Open source Tools

1. Java Path Finder- which implements the concolic testing strategy developed at NASA. (https://github.com/javapathfinder)
2. PIT test - mutation testing tool developed by Henry Coles. (http://pitest.org/)
3. S3 Constraint Solver developed by Microsoft