PIT Mutation Tool Additions

Matthew Bachelder
University of Texas at Dallas
United States
mtb140130@utdallas.edu

Vaishnavi Bhosale
University of Texas at Dallas
United States
vbb160030@utdallas.edu

Richard Fisher
University of Texas at Dallas
United States
rxf120030@utdallas.edu

ABSTRACT

Mutation testing is a method of software quality control that tests the unit test suite defined for programs by artificially injecting errors, or mutations, into software. Tests are then executed to evaluate the quality of the test suite. If the mutations are detected, then the mutation is killed and the test suite is deemed adequate with respect to that section of code. If the mutation is not detected, then it is said to have passed, and the test suite for that section of code is inadequate.

CCS CONCEPTS

• **Software engineering → Software testing and debugging**;

KEYWORDS

PIT Mutation testing, Mutators, Operators

1 INTRODUCTION

Mutation testing is a method of software quality control that tests the unit test suite defined for programs by artificially injecting errors, or mutations, into software. Tests are then executed to evaluate the quality of the test suite. If the mutations are detected, then the mutation is killed and the test suite is deemed adequate with respect to that section of code. If the mutation is not detected, then it is said to have passed, and the test suite for that section of code is inadequate.

Mutations, or faults that are injected into code, are modifications to various operators within the code. There are numerous mutations that can be used but commonly used examples include:

- Modification of increment operators to decrement
- Negation of certain values
- Modification of constants
- Modification of comparison operators

2 Proposed Additions to PIT

This project will add several mutations to the suite of available code modifications. These include:

a. ABS: Replaces a variable by its negation, e.g., a becomes –a
b. OBBN: Replaces the operators & by | and vice versa, e.g., a&b becomes a|b
c. AOD: Replaces an arithmetic expression by one of the operand, e.g., a + b becomes a
d. ROR: Replaces the relational operators with another one. It applies every replacement, e.g., < becomes ≥, or > becomes ≤
e. AOR: Replaces an arithmetic expression by another one. a + b becomes a ∗ b
f. UOI: Replaces a variable with a unary operator or removes an instance of an unary operator. a becomes a++
g. CRCR: Replaces a constant a with its negation, or with 1, 0, a + 1, a – 1, e.g., a becomes −a, and a becomes a − 1.

3 PIT: REAL WORLD MUTATION

PIT is a mutation testing framework for Java developed to support the day to day development on real codebases. This means that PIT aims at: (1) having a good integration with build tools like Maven, Ant, Gradle and integrated development environments like Eclipse or IntelliJ. (2) being fast: PIT uses three techniques to obtain its results: working on bytecode instead of source code, selecting the tests to run against the mutants and minimizing the number of mutant executions. (3) making a clear report of the tests execution. This makes the navigation between source code and mutants easy by highlighting mutants that were not killed.

4 Experimental Evaluation

The proposed modifications to PIT will be evaluated on five real world projects chosen from GitHub. Each project will contain a minimum of 1,000 lines of code, and will be evaluated after running at least 50 tests per project. The overall quality of the test suites, when tested with the augmented PIT, will be measured using the mutation adequacy score given in equation (1).

$$MAS\ (P, TS) = \frac{\text{Kn}}{M - En} \qquad (1)$$

Where, P = Program under test, TS = Test suite, Kn = number of Killed mutants, M = total number of Mutants, En = number of equivalent mutants.

REFERENCES

[1] [11] H. Coles. 2017. "PIT." Retrieved from http://pitest.org/.

[2] Coles, Henry, Thomas Laurent, Christopher Henard, Mike Papadakis, and Anthony Ventresque. "PIT: a practical mutation testing tool for Java." In Proceedings of the 25th International Symposium on Software Testing and Analysis, pp. 449-452. ACM, 2016.

[3] Rani, Shweta, et al. "Experimental Comparison of Automated Mutation Testing Tools for Java." *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, 2015, doi:10.1109/icrito.2015.7359265.