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

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics • **Networks** → Network reliability

KEYWORDS

ACM proceedings, text tagging

ACM Reference format:

G. Gubbiotti, P. Malagò, S. Fin, S. Tacchi, L. Giovannini, D. Bisero, M. Madami, and G. Carlotti. 1997. SIG Proceedings Paper in word Format. In *Proceedings of ACM Woodstock conference, El Paso, Texas USA, July 1997 (WOODSTOCK’97)*, 4 pages.

DOI: 10.1145/123 4

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.

PIT is a mutation mutation testing system that allows test coverage measurement of a test suite. 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.

REFERENCES

|  |  |
| --- | --- |
| [1] | [11] H. Coles. 2017. “PIT.” Retrieved from http://pitest.org/. |