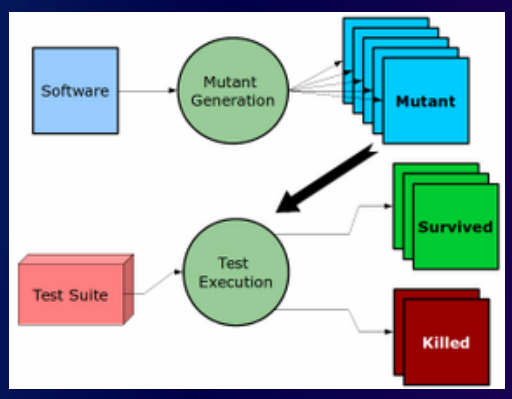
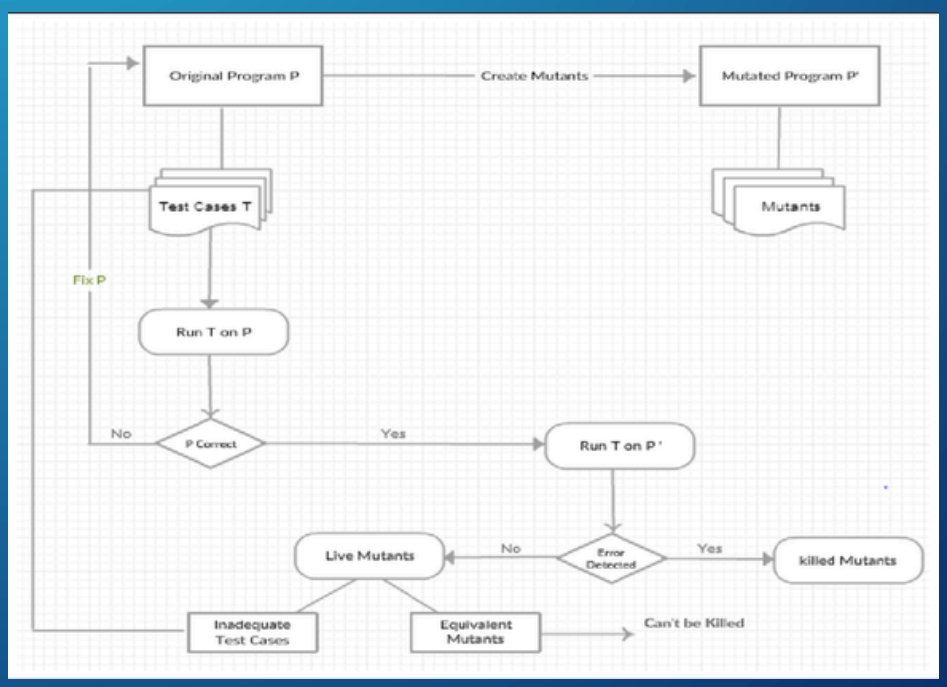
Mutation Testing

Karan 16010421072
Keyur 16010421073
Tirth 16010421075





INTRODUCTION

Mutation testing is a technique used to evaluate the effectiveness of test cases by intentionally introducing small modifications (mutants) in the code. The objective is to verify whether the existing tests can detect these modifications. If a mutant is detected by failing a test, it is "killed"; if not, the test suite is considered weak.

- **Original Program (P):** The starting point is the original program, and test cases (T) are designed to test it.
- **Create Mutants:** Small modifications (mutants) are made to the original program, resulting in a mutated program (P').
- **Run Test Cases on P:** The test cases are run on the original program to check its correctness.
- **Run Test Cases on P':** The same test cases are then executed on the mutated program.
- **Live Mutants Analysis.**

Types of Mutation Testing

- **Statement Mutation Testing -** This involves modifying entire statements or blocks of code. It changes how a certain statement executes within the program flow.
- **Logical Mutation Testing -** Focuses on changing logical operators within conditional statements. This type of mutation aims to see if a test case can detect a change in the logic.
- **Value Mutation Testing -** Involves changing literal values or constants in the code to see if test cases can detect these changes.

```python
#original code
def calculate_total(price, discount):
    if discount > 0:
        return price - discount
    return price


#mutated version
def calculate_total(price, discount):

    return price - discount
```

snappify.com

Statement mutation

```python
#original code
def is_eligible_for_voting(age):
    return age ⩾ 18


#mutated version
def is_eligible_for_voting(age):
    # Mutation: Changed '⩾' to '>'
    return age > 18
```

snappify.com

Logical mutation

When to Use Mutation Testing?

- **Critical Software:** Use mutation testing in applications where software reliability is critical, such as in healthcare, aviation, or financial systems.
- **Code Reviews:** It can be integrated into the code review process to ensure that changes do not introduce weak tests.
- **Continuous Integration (CI):** Mutation testing can be part of the CI pipeline to maintain high code quality over time.

ADVANTAGES

Some of the Advantages for Mutation testing are:

- **Improves Test Suite Effectiveness:** Mutation testing identifies weaknesses in the test suite, ensuring that it can catch subtle bugs by highlighting missed cases.
- **Helps Identify Redundant Test Cases:** It allows developers to recognize and remove test cases that do not contribute to error detection, simplifying the test suite.
- **Encourages Better Test Case Design:** By revealing untested scenarios, mutation testing encourages developers to create more thoughtful and comprehensive test cases.

INDUSTRY TRENDS

Industry Applications for mutation testing:

1. Defense Systems

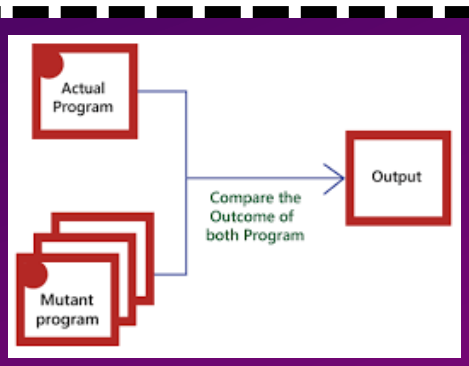
- **Application:** Used for software controlling navigation and targeting.
- **Example:** Mutating algorithms for trajectory calculations to simulate potential errors, ensuring robust response to real-world anomalies.

2. Healthcare Systems

- **Application:** Ensures the accuracy of medical software in devices and patient management systems.
- **Example:** Altering dosage calculations in drug delivery systems to verify that test cases can catch incorrect dosages.

References

- https://www.geeksforgeeks.org/software-testing-mutation-testing/
- https://www.researchgate.net/publication/310773886_Mutation_Testing_Techniques_A_Comparative_Study
- https://www.javatpoint.com/mutation-testing



Somaiya TRUST

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering