Mutation Testing (with Stryker Mutator)

Raoul Adler, 10. Sept. 2021

Overview

1. Repetition – What is mutation testing?
2. Stryker Mutator – A Mutation Testing framework
3. Pros and Cons

What is Mutation Testing?

- mainly used for Unit Testing
- is a type of White Box Testing
- changes/mutates certain statements of the source code
- checks if the test cases are able to find errors in source code

Code coverage versus Mutation Testing (I)

- Code coverage describes the degree of source code that a program executes when running a test suite

```
=============================== Coverage summary ===============================
Statements   : 100% ( 14/14 )
Branches     : 100% ( 4/4 )
Functions    : 100% ( 5/5 )
Lines        : 100% ( 13/13 )
================================================================================
```

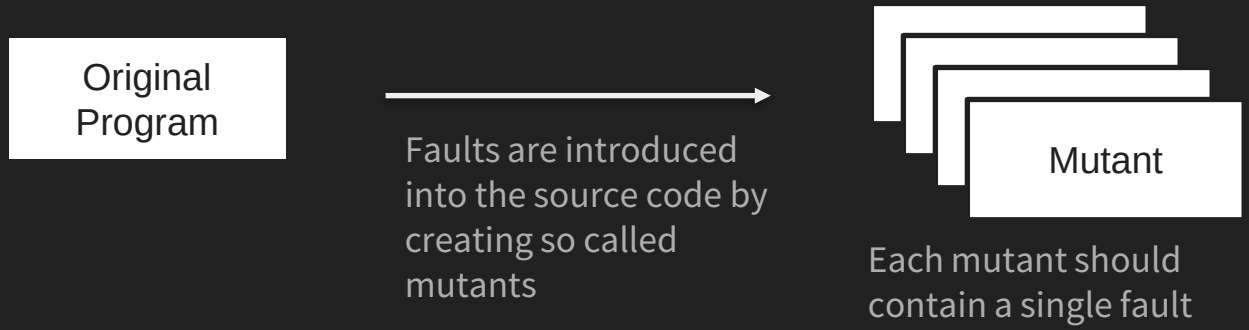
Code coverage versus Mutation Testing (II)

- Mutation testing measures the test suite effectiveness

File	% score	# killed	# timeout	# survived	# no cov	# error
All files	92.86	13	0	1	0	0
order.service.ts	92.86	13	0	1	0	0

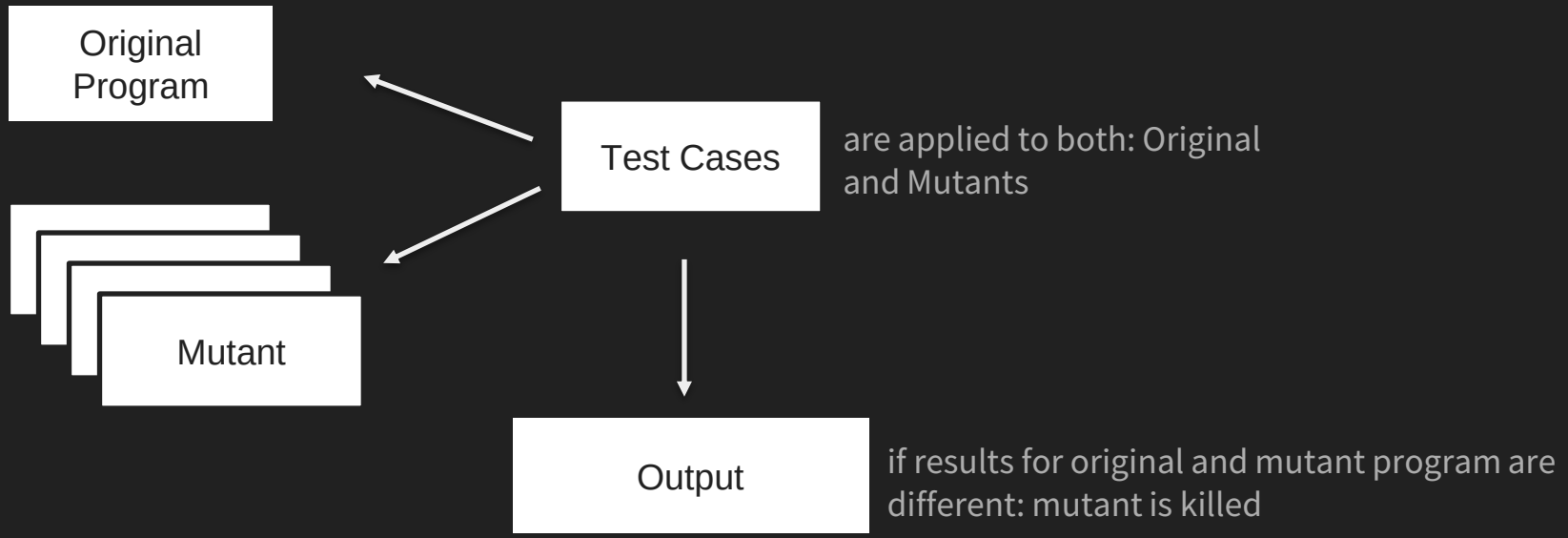
Goal of Mutation Testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code

How does Mutation testing work? (I)



The goal is to cause the mutant version to fail which demonstrates the effectiveness of the test cases.

How does Mutation testing work? (II)



What is a Mutant?

- is nothing but a single syntactic change that is made to a program statement
- each mutant program should differ from the original program by one mutation

Examples of Mutators

Original	Mutant
!=	==
==	!=
a==b	true
+	-
/	*
return new Object()	return null

https://stryker-mutator.io/docs/mutation-testing-elements/supported-mutators

survived means that a mutant in the original code did not cause a test to fail
survived means that a mutant in the original code did not cause a test to fail

Mutant Status

killed: a mutant in the original code caused a test to fail; the mutant is dead!

survived: a mutant in the original code did not cause a test to fail

Example

```
isALessB(a: number, b: number) {
  if (a < b) {
    return true;
  }
  return false;
}
```

Original

code coverage 100%

```
it( expectation: 'should be true', assertion: () => {
  const result = service.isALessB( a: 1, b: 2);
  expect(result).toBeTruthy();
});
```

Test

```
it( expectation: 'should be false', assertion: () => {
  const result = service.isALessB( a: 2, b: 1);
  expect(result).toBeFalsy();
});
```

tests passe
mutant survives

1 < 2 => true
2 > 1 => false

```
isALessB(a: number, b: number) {
  if (a <= b) {
    return true;
  }
  return false;
}
```

Mutant

test fails
mutant killed

expect: result is false
actual: result is true (1 <= 1)

```
it( expectation: 'should be false...', assertion: () => {
  const result = service.isALessB( a: 1, b: 1);
  expect(result).toBeFalsy();
});
```

+ Testcase

Stryker - A framework for mutation testing

Stryker – in a nutshell

- Supports C#, Scala, JavaScript and TypeScript
- Test runner agnostic
- > 30 supported mutators
- Open source
- For more info: https://stryker-mutator.io

Stryker In Action

Let’s do some demo!

Advantages of Mutation Testing

- is a powerful approach to attain high coverage of the source program
- has the capacity to detect all the faults in the program
- customers are benefited from this testing by getting a most reliable and stable system

Disadvantages of Mutation Testing

- is extremely costly and time-consuming since there are many mutant programs that need to be generated

- since its time consuming, this testing cannot be done without an automation tool

Thank you for your attention!

References

- https://www.guru99.com/mutation-testing.html#1
- http://www.diva-portal.org/smash/get/diva2:1161156/FULLTEXT01.pdf
- https://stryker-mutator.io/
- https://www.methodpark.de/blog/mutation-testing-oder-wie-gut-sind-meine-tests-wirklich/