Mutation Testing

What / Why / How

michal.smilowski@css.ch - CSS Insurance

What is mutation testing?

White box

Introduce Mutations to program code

Mutant output should vary from original output

What is mutation testing?

White box

Introduce Mutations to program code

Mutant output should vary from original output

Check effectiveness of test cases

Good test cases fail on mutated code

What is a mutation?

LOW LEVEL
CODE MODIFICATIONS

Value Mutations		
Change values / constants occurring in the programm Small ⟶ Large vs. Large ⟶ Small		

What is a mutation?

LOW LEVEL CODE MODIFICATIONS

Value Mutations	Decision Mutations	
Change values / constants occurring in the programm Small ⟶ Large vs. Large ⟶ Small	Typically mutations to logical operators: AND, OR, XOR, NOT etc. or arithmetic operators: (+)⟶ (-) (*) ⟶ (**) (+)⟶(i++)	

What is a mutation?

LOW LEVEL
CODE MODIFICATIONS

Value Mutations	Decision Mutations	Statement Mutations
Change values / constants occurring in the programm Small → Large vs. Large → Small	Typically mutations to logical operators: AND, OR, XOR, NOT etc. or arithmetic operators: (+)→ (-) (*) → (**) (+)→(i++)	Removing or replacing lines / statements in code

Value Mutation

Original Code

```
def square():
    return pow(a, 2)
```

Mutated Code

```
def square():
    return pow(a, 20000)
```

Decision Mutation

Original Code

```
def subtract(self, a, b):
        return a - b
```

Mutated Code

```
def subtract(self, a, b):
        return a / b
```

Statement Mutation

Original Code

```
def some_logic(self, a, b):
    if (a > b):
        r = 15
    else:
        r = 10
```

Mutated Code

```
def some_logic(self, a, b):
    if (a > b):
        S = 15
    else:
        S = 10
```

mutmut - python mutation tester

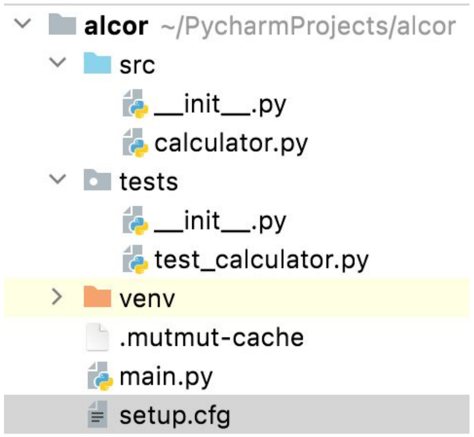
Hands on experience

```python
class Calculator:

    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

    def multiply(self, a, b):
        return a * b

    def divide(self, a, b):
        if b == 0:
            raise ArithmeticError()
        return a / b

    def square(self, a):
        return pow(a, 2)

    def some_logic(self, a, b):
        if a > b:
            return 15
        else:
            return 10
```

```python
class TestCalculator(TestCase):
    calculator = Calculator()

    def test_add(self):
        result = self.calculator.add(1, 2)
        self.assertEquals(3, result)

    def test_subtract(self):
        result = self.calculator.subtract(1, 2)
        self.assertEquals(-1, result)

    def test_multiply(self):
        result = self.calculator.multiply(1, 2)
        self.assertEquals(2, result)

    def test_divide(self):
        result = self.calculator.divide(1, 1)
        self.assertEquals(1, result)

    def test_square(self):
        result = self.calculator.square(5)
        self.assertEquals(25, result)

    def test_some_logic(self):
        result = self.calculator.some_logic(6, 5)
        self.assertEquals(15, result)
```

mutmut - python mutation tester

Project setup



```
alcor ~/PycharmProjects/alcor
  src
    __init__.py
    calculator.py
  tests
    __init__.py
    test_calculator.py
  venv
  .mutmut-cache
  main.py
  setup.cfg
```

1. Install package

```
% pip3 install mutmut
```

2. Configure - setup.cfg

```
[mutmut]
paths_to_mutate=src/
backup=False
tests_dir=tests/
```

mutmut - python mutation tester

Running mutation tests

```
% mutmut run

Legend for output:
🎉 Killed mutants.   The goal is for everything to end up in this bucket.
⏰ Timeout.          Test suite took 10 times as long as the baseline so were killed.
🤔 Suspicious.       Tests took a long time, but not long enough to be fatal.
🙁 Survived.         This means your tests need to be expanded.
🔇 Skipped.          Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
⠸ 11/11  🎉 10  ⏰ 0  🤔 0  🙁 1  🔇 0
```

mutmut - python mutation tester

Running mutation tests

```
% mutmut run

Legend for output:
🎉 Killed mutants.   The goal is for everything to end up in this bucket.
⏰ Timeout.          Test suite took 10 times as long as the baseline so were killed.
🤔 Suspicious.       Tests took a long time, but not long enough to be fatal.
🙁 Survived.         This means your tests need to be expanded.
🔇 Skipped.          Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
⠏ 11/11  🎉 10  ⏰ 0  🤔 0  🙁 1  🔇 0
```

```
% mutmut show src/calculator.py

 Survived 🙁 (1)

 ---- src/calculator.py (1) ----

 # mutant 43
 --- src/calculator.py
 +++ src/calculator.py
 @@ -19,7 +19,7 @@
          return pow(a, exp)

      def some_logic(self, a, b):
 -        if a > b:
 +        if a >= b:
              return 15
          else:
              return 10
```

mutmut - python mutation tester

Running mutation tests

```
% mutmut run

Legend for output:
🎉 Killed mutants.   The goal is for everything to end up in this bucket.
⏰ Timeout.          Test suite took 10 times as long as the baseline so were killed.
🤔 Suspicious.       Tests took a long time, but not long enough to be fatal.
🙁 Survived.         This means your tests need to be expanded.
🔇 Skipped.          Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
⠏ 11/11  🎉 10  ⏰ 0  🤔 0  🙁 1  🔇 0
```

```
class TestCalculator(TestCase):
    calculator = Calculator()


    def test_some_logic(self):
        result = self.calculator.some_logic(4, 5)
        self.assertEquals(10, result)
```

```
% mutmut show src/calculator.py

 Survived 🙁 (1)

 ---- src/calculator.py (1) ----

 # mutant 43
 --- src/calculator.py
 +++ src/calculator.py
 @@ -19,7 +19,7 @@
          return pow(a, exp)

     def some_logic(self, a, b):
 -        if a > b:
 +        if a >= b:
              return 15
          else:
              return 10
```

mutmut - python mutation tester

Test improvements

```
% mutmut run

Legend for output:
🎉 Killed mutants.   The goal is for everything to end up in this bucket.
⏰ Timeout.          Test suite took 10 times as long as the baseline so were killed.
🤔 Suspicious.       Tests took a long time, but not long enough to be fatal.
🙁 Survived.         This means your tests need to be expanded.
🔇 Skipped.          Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
⠋ 11/11  🎉 10  ⏰ 0  🤔 0  🙁 0  🔇 0
```

```
class TestCalculator (TestCase):
    calculator = Calculator()


    def test some logic (self):
        result = self.calculator.some _logic( 4, 5)
        self.assertEquals( 10, result)

    def test some logic2 (self):
        result = self.calculator.some _logic( 5, 5)
        self.assertEquals( 10, result)
```

Pros & Cons

+ Brings attention to new types of errors
+ Increases coverage of tested application / code
+ Introduces edge cases testing

- Can be resources and time consuming
- Not suited for black box testing
- Setup + automation required

Thanks / Q&A

Michal Smilowski
Software engineer @ CSS Insurance

✉ michal.smilowski@css.ch
msmilowski

Sources:

1. https://www.javatpoint.com/mutation-testing
2. https://mutmut.readthedocs.io/en/latest/index.html#example-mutations
3. https://techbeacon.com/app-dev-testing/mutation-testing-how-test-your-tests