

Análise e Teste de Software

Mutation Testing

Universidade do Minho

2022/2023

1 Mutação de Código

A mutação de código permite avaliar a qualidade dos testes. Caso se mude certa parte do código e nenhum teste passe a falhar por causa disso, então é possível que erros estejam naquela parte do código e que não sejam detectados pelos testes. Por exemplo, o seguinte código tem um erro:

```
public class Calculadora {
    int resultado;
    public Calculadora() {
        resultado = 0;
    }
    int adiciona(int y) {
        resultado = resultado + y;
        return resultado;
    }
    int adiciona(int x, int y) {
        resultado = x * y;
        return resultado;
    }
    int subtrai(int y) {
        resultado = resultado - y;
        return resultado;
    }
    int subtrai(int x, int y) {
        resultado = x - y;
        return resultado;
    }
    int ultimoResultado() {
        return resultado;
    }
}
```

mas os seguintes testes, mesmo tendo cobertura total do método com erro, não o encontram:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class CalculadoraTest {
    @Test
    public void adiciona01() {
        Calculadora c = new Calculadora();
        int x = c.adiciona(5);
        assertEquals(5, x);
    }
    @Test
    public void adiciona02() {
        Calculadora c = new Calculadora();
        int x = c.adiciona(2, 2);
        assertEquals(4, x);
    }
    @Test
    public void subtrai01() {
        Calculadora c = new Calculadora();
        int x = c.subtrai(0, 0);
        assertEquals(0, x);
    }
}
```

A execução manual de mutações é lenta e ineficiente. Para resolver este problema, existem programas que geram mutações automaticamente e verificam se os testes atuais são suficientes. Um desses programas é o *PIT* ¹. Podemos utilizar esta ferramenta incorporando-a na configuração *Maven* do nosso projeto, ou utilizando um plugin *IntelliJ* que permite uma utilização mais simples deste programa.

A instalação do plugin do *IntelliJ* deverá ser bastante simples: File -> Settings -> Plugins -> Pit Mutation Testing -> Install. Feita a instalação, basta fazer clique com o lado direito do rato na classe a testar, sendo que deverão existir duas novas opções relacionadas com o *PIT*. Também é possível preparar uma configuração mais detalhada do tipo de execução *PIT* a fazer no projeto.

A utilização de *PIT* com *Maven* está detalhada no seguinte link: <https://pitest.org/quickstart/maven/>. Resumidamente, deverá bastar acrescentar o plugin à configuração e depois executar a goal correta.

¹<https://pitest.org/>

2 Exercícios

1. Crie um projeto novo no *IntelliJ*. Irá colocar nesse projeto apenas as classes *Calculadora* e *CalculadoraTest*. Tente resolver os exercícios desta alínea utilizando *IntelliJ* mas não *Maven*.
 - (a) Encontre o erro no código apresentado acima e defina um teste que consiga detectar o erro.
 - (b) Corrija o erro e execute a bateria de testes. Verifique que todos os testes passam.
 - (c) Utilize o *PIT* para testar e melhorar a sua bateria de testes:
 - i. Clique com o lado direito do rato na classe *Calculadora* e faça Pit test desta.
 - ii. Se a execução tiver sucesso, irá ser gerado um ficheiro HTML na pasta do projeto e será colocado um link para esse ficheiro no terminal do *IntelliJ*. Abra esse ficheiro e analise o relatório.
 - iii. Analise a percentagem de mutações que sobrevivem aos seus testes. O que quer dizer este número? Acrescente mais testes unitários por forma a reduzir ao máximo possível a quantidade de mutações que sobrevivem.
2. Volte ao projeto com as classes *Aluno* e *Turma* utilizado nas aulas anteriores. Tente resolver os exercícios desta alínea utilizando *Maven* mas não *IntelliJ* (excepto como editor de código básico).
 - (a) Altere o seu ficheiro de configuração *pom.xml* para utilizar o *PIT*.
 - (b) Utilize o *PIT* para testar e melhorar a sua bateria de testes:
 - i. Descubra qual o comando *Maven* a executar para gerar o relatório *PIT*, e utilize-o. Se considerar útil, considere criar uma *Makefile* com os comandos *Maven* que tem vindo a ser utilizados nas últimas aulas.
 - ii. Se a execução tiver sucesso, irá ser gerado um ficheiro HTML na pasta do projeto. Abra esse ficheiro e analise o relatório.
 - iii. Analise a percentagem de mutações que sobrevivem aos seus testes. O que quer dizer este número? Acrescente mais testes unitários por forma a reduzir ao máximo possível a quantidade de mutações que sobrevivem.
 - (c) (Extra) Manualmente produza 5 variações da classe *Turma*, cada um com um bug / mutação injetada manualmente. Corra a sua bateria de testes original e verifique se estes testes são capazes de apanhar o bug injetado.