Teste de Mutação

Prof. André Takeshi Endo

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

Teste de Mutação

- Também conhecido com um critério de teste
  - **Análise de mutantes**
- Um critério de teste da técnica caixa-branca
  - Técnica baseada em defeitos

Hipóteses

- **Programador competente**
  - Programador desenvolve um código muito próximo do correto
  - A correção é, em geral, uma pequena modificação
- **Efeito do acoplamento**
  - Casos de teste que revelam defeitos mais simples também revelam defeitos complexos
  - Um defeito complexo pode ser representado pela combinação de vários defeitos mais simples

Operador de Mutação

- Faz uma pequena alteração sintática
  - simular um defeito

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

**Exemplo adaptado de [DMJ07]

Operador de Mutação

- Faz uma pequena alteração sintática
  - simular um defeito

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Operador ORRN

ORRN
relational operator replacement

Troca um operador relacional por outros

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

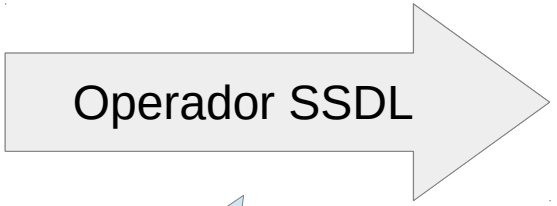
**Exemplo adaptado de [DMJ07]

Operador de Mutação

- Faz uma pequena alteração sintática
  - simular um defeito

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```



SSDL
statement deletion

Um determinado comando é eliminado do código

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

**Exemplo adaptado de [DMJ07]

Operador de Mutação

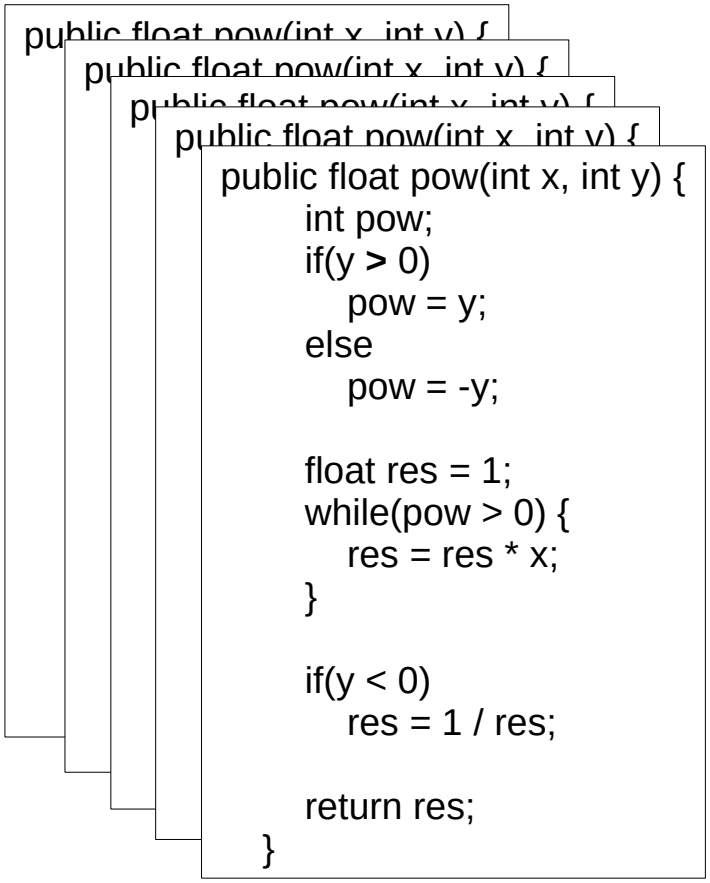
- Usando operadores de mutação, milhares de mutantes podem ser gerados

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Operadores de mutação

Mutantes



**Exemplo adaptado de [DMJ07]

Como calcular a cobertura?

- Escore de mutação
  - **(Núm. mutantes mortos) / (total de mutantes – núm. mutantes equivalentes)**
- **O que é um mutante equivalente?**
  - Um mutante é equivalente quando não existe um caso de teste capaz de diferenciá-lo do original
- **O que é um mutante morto?**
  - Quando um CT é capaz de diferenciar o mutante do original
  - Ou seja, o CT passa para o original e falha para o mutante

Como calcular a cobertura?

- Escore de mutação define um % semelhante aos critérios todos-nós e todas-arestas
- Se seus casos de teste são capazes de revelar esses defeitos artificiais (*matar os mutantes*), os casos de teste devem ser bons para revelar também defeitos reais
- Usar para melhorar a qualidade dos seus testes

Como matar um mutante?

- Escreve um CT que mate o mutante abaixo!
  - Passa no original e falha no mutante

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res + x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Como matar um mutante?

- Escreve um CT que mate o mutante abaixo!
  - Passa no original e falha no mutante

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res + x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

```
assertEquals(27.0, o.pow(3, 3), 0.0001);
```

```
assertEquals(27.0, o.pow(3, 3), 0.0001);
```

Como matar um mutante?

- Escreve um CT que mate o mutante abaixo!
  - Passa no original e falha no mutante

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res + x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

```
assertEquals(27.0, o.pow(3, 3), 0.0001);
```

```
assertEquals(27.0, o.pow(3, 3), 0.0001);
```

Como matar um mutante?

- Escreve um CT que mate o mutante abaixo!
  - Passa no original e falha no mutante

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

```
assertEquals(27.0, o.pow(3, 3), 0.0001);
```

Este mutante está morto!

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res + x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

```
assertEquals(27.0, o.pow(3, 3), 0.0001);
```

Mutantes Vivos e Equivalentes

- Um mutante está vivo quando os casos de teste atuais não o diferencia do original
  - Possui adicionar um CT capaz de matá-lo
- Um mutante é equivalente quando ***não existe um caso de teste*** capaz de diferenciá-lo do original
  - Marco este mutante e sigo em frente

Ferramentas para Java

- MuJava
  - https://cs.gmu.edu/~offutt/mujava/
- MuClipse
  - http://muclipse.sourceforge.net/
- **PITEST (recomendado)**
  - http://pitest.org/
  - No Eclipse, help → Eclipse Marketplace → Find → “pitest”
  - https://github.com/pitest/pitclipse
- Jester
  - http://jester.sourceforge.net/
- Major Mutation framework
  - http://mutation-testing.org/

Como matar um mutante?

- Escreve um CT que mate o mutante abaixo!
  - Passa no original e falha no mutante

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y > 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Exercícios

- Mate o mutante abaixo, se não for equivalente

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Exercícios

- Mate o mutante abaixo, se não for equivalente

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res + x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Exercícios

- Mate o mutante abaixo, se não for equivalente

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow != 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Exercícios

- Mate o mutante abaixo, se não for equivalente

Programa Original

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0)
            res = 1 / res;

        return res;
    }
```

Mutante

```
public float pow(int x, int y) {
        int pow;
        if(y >= 0)
            pow = y;
        else
            pow = -y;

        float res = 1;
        while(pow > 0) {
            res = res * x;
            pow--;
        }

        if(y < 0);



        return res;
    }
```

Exercícios

- Resolução em :
- https://github.com/andreendo/software-testing-undergrad-course/blob/master/codigos_cx_branca/test/exemplo/mutantes/MatandoMutantesTest.java

Bibliografia

- [Pfleeger07] S. L. Pfleeger, “Engenharia de Software: Teoria e Prática”, 2007.
- [Pressman11] R. S. Pressman, “Engenharia de Software: uma abordagem profissional”, 2011.
- [Sommerville03] I. Sommerville, “Engenharia de Software”, 2003.
- [Brooks87] “No Silver Bullet: Essence and Accidents of Software Engineering”, 1987.
  http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1663532
- [IEEE90] “IEEE Standard Glossary of Software Engineering Terminology”, 1990.
  http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=159342

Bibliografia

- [Myers] G. J. Myers, T. Badgett, C. Sandler, “The art of software testing”, 2012.
- [Pezze] M. Pezze, M. Young, “Teste e análise de software: Processos, princípios e técnicas”, 2008.
- [DMJ07] DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mario. Introdução ao teste de software. Rio de Janeiro, RJ: Elsevier, 2007. 394 p. ISBN 9788535226348.
- [UUU] Materiais didáticos elaborados pelos grupos de engenharia de software do ICMC-USP, DC-UFSCAR e UTFPR-CP.