

Teste de Software

Teste de Mutação: Critério de Teste Baseado em Erros

Lesandro Ponciano

Objetivos da Aula

- Contextualizar a técnica de teste baseada em erros
- Definir “mutantes” em Teste de Software
- Introduzir as etapas do Teste de Mutação

Técnica de Teste Baseada em Erros

- Há um conjunto de **erros frequentemente cometidos** durante o processo de desenvolvimento do software
- Os **erros mais comuns** podem ser utilizados como informação para teste do software
- Casos de teste baseados em erros frequentes

Mutantes

- Um código correto pode ter diversas versões incorretas
 - Cada versão é considerada um **mutante**
 - Cada mutante contém apenas um defeito
 - Uma mutação é uma simples mudança sintática
- Casos de teste são aplicados ao programa original e ao programa mutante
 - Para causar o programa mutante falhar
 - Assim, demonstrar a eficácia do caso de teste

Exemplo de Mutante

```
int max(int x, int y)
{
    int mx = x;
    if (x > y)
        mx = x;
    else
        mx = y;
    return mx;
}
```

Principal

```
int max(int x, int y)
{
    int mx = x;
    if (x < y)
        mx = x;
    else
        mx = y;
    return mx;
}
```

Mutante

Teste de Mutação

- É uma técnica que visa medir o quão apropriado é um conjunto de casos de teste
- Deve ser usado em conjunto com as estratégias de derivação de casos de teste

Hipóteses do Teste de Mutação

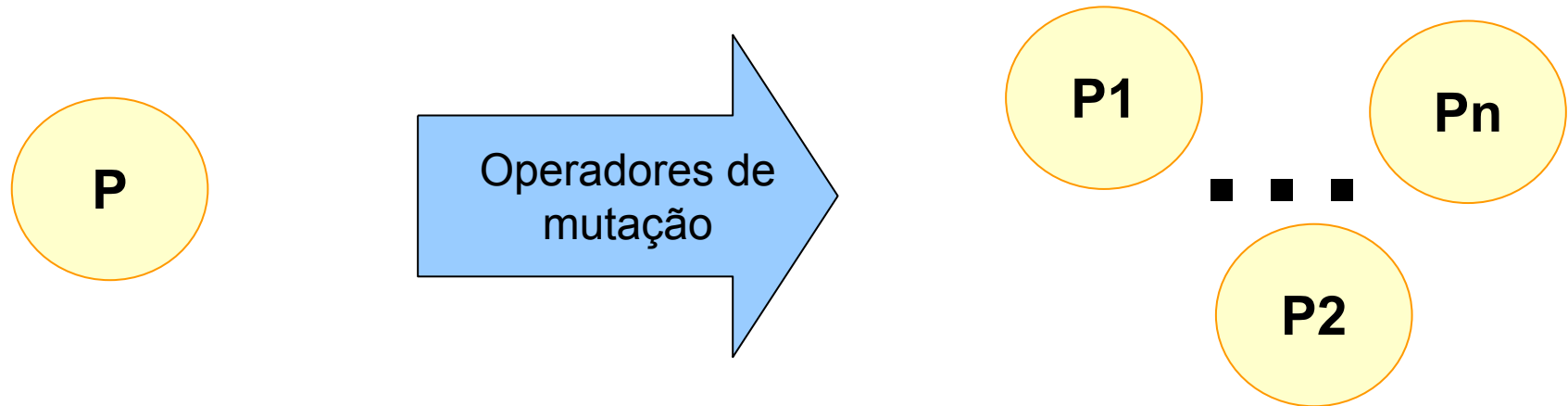
- **Programador Competente**
 - Programadores experientes escrevem programas corretos
 - Ou ele escrevem programas que, com poucas alterações, se tornam corretos
- **Efeito de Acoplamento**
 - Defeitos complexos estão relacionados a defeitos simples
 - Ou, se cada defeito simples for consertado, pode-se assumir que os defeitos complexos também o serão

Passos do Teste de Mutação

- 1) Geração de Mutantes
- 2) Execução do Programa
- 3) Execução dos Mutantes
- 4) Análise dos Mutantes Vivos

1 – Geração dos Mutantes

- Modelar os desvios sintáticos mais comuns
 - Aplicação de operadores de mutação em um programa
 - Gerando programas similares: mutantes



Programa em teste

Mutantes

Categorias de Operadores de Mutação

- Operador de mudança de operando
 - $if(5 < y) \rightarrow if(y < 5) \rightarrow if(y < x)$
- Operador de modificação de expressão
 - $if(x == y) \rightarrow if(x >= y) \rightarrow if(x < y)$
- Operador de modificação de comando
 - Deletar *else*, parte do *if-else*
 - Deletar todo o *if-else*
 - Inclusão de um *return*
- Depende muito da linguagem de programação

Operadores de Mutação em C

- Eliminação de comandos (SSDL, *Statement Deletion*)
- Troca de Operador Relacional (ORRN, *Relational operator replacement*)
- Armadilha em condição de comando if (STRI, *trap in if condition*)
- Troca de variáveis escalares (Vsrr, *scalar variable reference replacement*)

2 – Execução do Programa

- Execução do programa com os casos de teste

3 - Execução dos Mutantes

- Execução dos mutantes com os casos de teste
- Análise dos tipos de mutantes detectados
 - mortos e vivos
- **Mutante Morto**
 - Gera saída diferente do programa original
- **Mutantes Vivos**
 - Gera saída igual à do programa original

4 - Análise dos Mutantes Vivos

- Mutantes **não-equivalentes**
 - Os casos de teste foram insuficientes para matar
 - Deve-se criar um novo caso de teste
- Mutantes **equivalentes**
 - Não podem ser mortos

Detecção de Mutantes Equivalentes

- Estratégias baseadas em otimização de compiladores e análise de fluxo de dados
 - 1) Detecção de Código não alcançável
 - 2) Propagação de constantes
 - 3) Propagação de invariantes
 - 4) Detecção de expressões comuns
 - 5) Detecção de invariantes de repetição
 - 6) *Hoisting e sinking*

OFFUTT, A. Jefferson; CRAFT, W. Michael. Using compiler optimization techniques to detect equivalent mutants. *Software Testing, Verification and Reliability*, v. 4, n. 3, p. 131-154, 1994.

Mutation Score

- Para um conjunto de casos de teste, o *mutation score* é o percentual de mutantes não-equivalentes que foram mortos
- $MutationScore = 100 * D/(N-E)$
 - D = mutantes mortos
 - N = número de mutantes
 - E = número de mutantes equivalentes
- Um conjunto de casos de teste adequado tem *mutationScore* de 100%

Avaliação do Teste de Mutação

- Estudos empíricos mostram que o teste de mutação é uma forma efetiva de avaliar um conjunto de casos de teste
- Gerar todos os mutantes e executar cada caso de teste em todos os mutantes é custoso
- A suíte de teste pode variar muito dependendo da linguagem de programação

Referências

- Nakagawa, Elisa Yumi. (2015) Teste de Software - Parte 2. https://edisciplinas.usp.br/pluginfile.php/317501/mod_resource/content/1/Aula08_TestesSoftware_Parte2.pdf
- Dependable Software Systems. Topics in Mutation Testing and Program Perturbation. SERG <https://slideplayer.com/slide/7421728/>
- Myers, Glenford J. et al (2004) "The Art of Software Testing." 2ed. New York, NY, USA: John Wiley & Sons.
- DELAMARO, Márcio; MALDONADO, José; JINO, Mario. Introdução ao teste de software. Elsevier Brasil, 2016. **(Capítulo 5)**

Teste de Software

Prof. Dr. Lesandro Ponciano

<https://orcid.org/0000-0002-5724-0094>