

# Teste Estrutural

Gilmar Ferreira Arantes  
- [gilmar.arantes@goias.gov.br](mailto:gilmar.arantes@goias.gov.br)

GPS - SCTI

Goiânia, 24 e 25 de Junho de 2019

# Sumário I

1 Teste Estrutural

2 Referências

# Conceitos Básicos

- Técnica Caixa Branca oposto da Técnica Caixa Preta.

# Conceitos Básicos

- Técnica Caixa Branca oposto da Técnica Caixa Preta.
  - Baseia-se nos caminhos internos, estrutura e implementação do produto em teste.

# Conceitos Básicos

- Técnica Caixa Branca oposto da Técnica Caixa Preta.
  - Baseia-se nos caminhos internos, estrutura e implementação do produto em teste.
  - **Requer conhecimento do código do produto em teste para ser aplicada.**

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.
  - 2 Caminhos através da implementação são escolhidos.



# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.
  - 2 Caminhos através da implementação são escolhidos.
  - 3 Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.
  - 2 Caminhos através da implementação são escolhidos.
  - 3 Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  - 4 As saídas esperadas para as entradas escolhidas são determinadas.

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.
  - 2 Caminhos através da implementação são escolhidos.
  - 3 Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  - 4 As saídas esperadas para as entradas escolhidas são determinadas.
  - 5 Os casos de testes são construídos.

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.
  - 2 Caminhos através da implementação são escolhidos.
  - 3 Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  - 4 As saídas esperadas para as entradas escolhidas são determinadas.
  - 5 Os casos de testes são construídos.
  - 6 Os casos de testes são executados.

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.
  - 2 Caminhos através da implementação são escolhidos.
  - 3 Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  - 4 As saídas esperadas para as entradas escolhidas são determinadas.
  - 5 Os casos de testes são construídos.
  - 6 Os casos de testes são executados.
  - 7 As saídas obtidas são comparadas com as saídas esperadas.

# Processo do Teste Caixa Branca

- Os passos básicos para se aplicar um critério de teste caixa branca são os seguintes:
  - 1 A implementação do produto em teste é analisada.
  - 2 Caminhos através da implementação são escolhidos.
  - 3 Valores de entradas são selecionados de modo que os caminhos selecionados sejam executados.
  - 4 As saídas esperadas para as entradas escolhidas são determinadas.
  - 5 Os casos de testes são construídos.
  - 6 Os casos de testes são executados.
  - 7 As saídas obtidas são comparadas com as saídas esperadas.
  - 8 Um relatório é gerado para avaliar o resultado dos testes.

# Critérios de Teste Estrutural

- Critérios Baseados em Fluxo de Controle:
  - Todos-Nós, Todas-Arestas, Todos-Caminhos.
- Critérios Baseados em Fluxo de Dados:
  - Todas-Definições, Todos-P-Usos, Todos-C-Usos, Todos-Usos, Todos-Du-Caminhos, Todos-Pot-Usos, Todos-Pot-Du-Caminhos, Todos-Pot-Usos/Du.

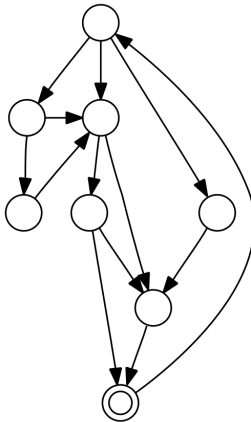
# Uso nas Fases de Teste

- Critérios da técnica caixa branca também podem ser utilizados em todas as fases/níveis de teste.
- São mais comuns no teste de unidade e de integração.
- Teste de caminhos:
  - Caminhos dentro de uma unidade.
  - Caminhos entre unidades.
  - Caminhos entre sub-sistemas.
  - Caminhos entre o sistema todo.



# Uso nas Fases de Teste

- Diferentes caminhos dentro de um produto em teste.



# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).

# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:

# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - $y = x * 2$ ; // deveria ser  $y = x ** 2$

# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - $y = x * 2$ ; // deveria ser  $y = x ** 2$
  - funciona corretamente para  $x = 0$ ,  $y=0$  e  $x = 2$ ,  $y = 4$ .

# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - $y = x * 2$ ; // deveria ser  $y = x ** 2$
  - funciona corretamente para  $x = 0$ ,  $y=0$  e  $x = 2$ ,  $y = 4$ .
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.

# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - $y = x * 2$ ; // deveria ser  $y = x ** 2$
  - funciona corretamente para  $x = 0$ ,  $y=0$  e  $x = 2$ ,  $y = 4$ .
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.
- **Determinação de caminhos não executáveis pode ser um problema.**

# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - $y = x * 2$ ; // deveria ser  $y = x ** 2$
  - funciona corretamente para  $x = 0$ ,  $y=0$  e  $x = 2$ ,  $y = 4$ .
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.
- Determinação de caminhos não executáveis pode ser um problema.
- **Dificuldade de automatização.**



# Desvantagens do Teste Caixa Branca

- O número de caminhos a serem executados pode ser infinito (semelhante ao teste exaustivo).
- O caso de teste selecionado pode não revelar o defeito sensível a dado. Por exemplo:
  - $y = x * 2$ ; // deveria ser  $y = x ** 2$
  - funciona corretamente para  $x = 0$ ,  $y=0$  e  $x = 2$ ,  $y = 4$ .
- Assume fluxo de controle correto (ou próximo do correto). Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.
- Determinação de caminhos não executáveis pode ser um problema.
- Dificuldade de automatização.
- Habilidades de programação avançadas exigidas para compreender o código e decidir pela executabilidade ou não de um caminho.

# Vantagens do Teste Caixa Branca

- Eficaz em determinar defeitos de lógica e de programação, especialmente no nível de unidade.
- É possível garantir que partes essenciais ou críticas do programa tenham sido executadas.
  - Requisito mínimo de teste: garantir que o programa foi liberado tendo seus comandos executados ao menos uma vez por pelo menos um caso de teste.
- É possível saber o que ainda não foi testado.

# Níveis de Cobertura

- Diferentes níveis de cobertura podem ser definidos em função dos elementos do GFC.
- Cobertura: porcentagem dos requisitos que foram testados versus o total de requisitos gerados.
- Oito diferentes níveis de cobertura são definidos por [Copeland, 2003].
- Quanto maior o nível, maior o rigor do critério de teste, ou seja, mais caso de teste ele exige para ser satisfeito.
  - Nível 0 ← Nível 1 ← Nível 2 ← Nível 3 ← Nível 4 ← Nível 5 ← Nível 6 ← Nível 7

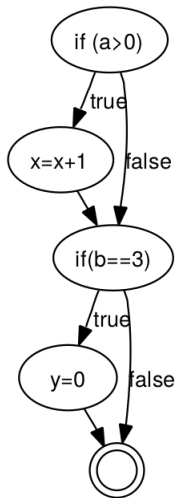
# Níveis de Cobertura

- Nível 0: qualquer valor de cobertura inferior a 100% da cobertura de todos os comandos.
- Nível 1: 100% de cobertura de comandos. (Todos-Nós)
- Nível 2: 100% de cobertura de decisões. (Todos-Arcos)
- Nível 3: 100% de cobertura de condições.
- Nível 4: 100% de cobertura decisões/condições.
- Nível 5: 100% de cobertura de condições múltiplas.
- Nível 6: cobertura de loop.
- Nível 7: 100% de cobertura de caminhos. (Todos-Caminhos)

# Níveis de Cobertura - Exemplo

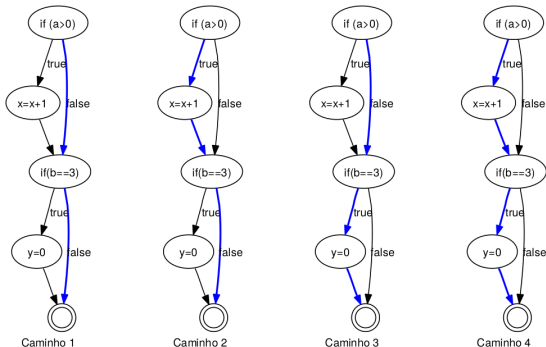
```

1  if (a>0){
2      x=x+1;
3  }
4  if (b==3){
5      y=0;
6  }
    
```






# Níveis de Cobertura - Exemplo

Nível 1: 100% de cobertura de comandos (requisito mínimo de teste)



Um caso de teste é suficiente para cobrir todos os comandos mas não todos os caminhos. Por exemplo, use  $a=6$  e  $b=3$  para cobrir o "Caminho 4".

# Referências Bibliográficas

-  B. Boehm and V. R. Basili.  
Software defect reduction top 10 list.  
*Computer*, 34(1):135-137, 2001. ISSN 0018-9162. doi:  
<http://dx.doi.org/10.1109/2.962984>.
-  Lee Copeland.  
A Practitioner's Guide to Software Test Design  
*Artech House Publishers, Boston, 2003*.
-  R. D. Craig and S. P. Jaskiel.  
Systematic Software Testing.  
*Artech House Publishers, 2002*.
-  Imagens, Google  
Google Imagens.  
<https://www.google.com.br>. Acesso em 19/06/2019.
-  IEEE.  
IEEE standard glossary of software engineering terminology.  
Standard 610.12-1990 (R2002),  
*IEEE Computer Society Press, 2002*.
-  Harley, Nick.  
11 of the most costly software errors in history [2019 update]  
<https://raygun.com/blog/costly-software-errors-history/>. Acesso em 19/06/2019.
-  Neto, Arilo Cláudio Dias Neto.  
Introdução ao Teste de Software.  
*Engenharia de Software Magazine. Ed. 01, 2009*
-  F. Shull, V. Basili, B. Boehm, A. W. Brown, P. Costa, M. Lindvall, D. Port, I. Rus, R.e Tesoriero, and M. Zelkowitz.  
What we have learned about fighting defects.  
*In VIII International Symposium on Software Metrics - METRICS'02, pages 249-258, Washington, DC, USA, June 2002. IEEE Computer Society. ISBN 0-7695-1339-5.*