

Técnica de Teste Caixa Branca

Critérios Baseados em Fluxo de Controle

Auri Marcelo Rizzo Vincenzi
Gilcimar Divino de Deus

Instituto de Informática
Universidade Federal de Goiás

22 de agosto de 2008



Critérios Baseados em Fluxo de Controle

Definições

Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

Passos do Teste de Caminho Básico

Complexidade Ciclomática

Criação do Conjunto de Caminhos

Exemplo

Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclométrica

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

Introdução

- ▶ Critérios pertencentes à Técnica de Teste Caixa Branca.
- ▶ Identificam requisitos de testes (caminhos de execução) a partir da implementação do produto em teste.
- ▶ Requer a criação e execução de casos de testes que exercitem tais requisitos.
- ▶ Definição:
 - ▶ Caminho: seqüência de execução de comandos que se inicia em um ponto de entrada e termina em um ponto de saída do produto em teste.

Exemplo de Problemas (1)

- ▶ Infelizmente, o teste exaustivo de todos os possíveis caminhos de fluxo de controle possui várias desvantagens:
 - ▶ O número de caminhos pode ser infinito ou muito grande. Cada decisão dobra e cada loop multiplica o número de caminhos.

```
1  for (i=1; i<=1000; i++)  
2      for (j=1; j<=1000; j++)  
3          for (k=1; k<=1000; k++)  
4              doSomethingWith(i, j, k);
```

executa `doSomethingWith()` um bilhão de vezes ($1000 \times 1000 \times 1000$).

- ▶ Caminhos presentes na especificação podem ser esquecidos na implementação:

```
1  if (a>0) dolsGreater();  
2  if (a==0) dolsEqual();  
3  // comando ausente — if (a<0) dolsLess();
```

Exemplo de Problemas (2)

- ▶ Mais problemas...

- ▶ Defeitos podem existir mesmo com o fluxo de controle correto.

```
1 // atual mas incorreto
2 a=a+1;
3 // código correto
4 //a=a-1;
```

- ▶ Um módulo pode executar corretamente para diversos casos de testes e falhar para alguns:

```
1 int blech (int a, int b) {
2     return a/b;
3 }
```

falha se b assumir o valor 0 mas executa corretamente se b for diferente de 0.

- ▶ Mesmo com tais limitações o teste caixa branca é de grande importância e complementar ao teste caixa preta.

Critérios Baseados em Fluxo de Controle

Definições

Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

Passos do Teste de Caminho Básico

Complexidade Ciclomática

Criação do Conjunto de Caminhos

Exemplo

Aplicabilidade e Limitações

Resumo

Exercício


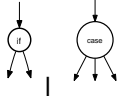

Leitura Recomendada

Referências

O que é?

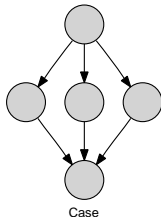
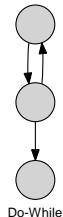
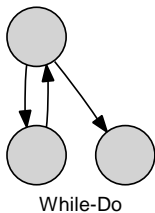
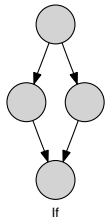
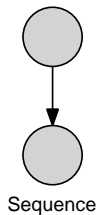
- ▶ Notação gráfica utilizada para abstrair o fluxo de controle lógico de um programa;
- ▶ Composto de nós e arcos;
- ▶ Um **nó** representa uma ou mais instruções as quais são sempre executadas em seqüência, ou seja, uma vez executada a primeira instrução de um nó todas as demais instruções daquele nó também são executadas;
- ▶ Um **arco**, também chamado de ramo ou aresta, representa o fluxo de controle entre blocos de comandos (nós).

Elementos presentes no GFC (1)

	grupo de comandos executados seqüencialmente do início ao fim.
	ponto de mudança de fluxo de controle.
	ponto no qual fluxos de controle são unidos.

Elementos presentes no GFC (2)

Para representar uma função ou um método como um GFC, as seguintes construções podem ser utilizadas:



Critérios Baseados em Fluxo de Controle

Definições

Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

Passos do Teste de Caminho Básico

Complexidade Ciclomática

Criação do Conjunto de Caminhos

Exemplo

Aplicabilidade e Limitações

Resumo

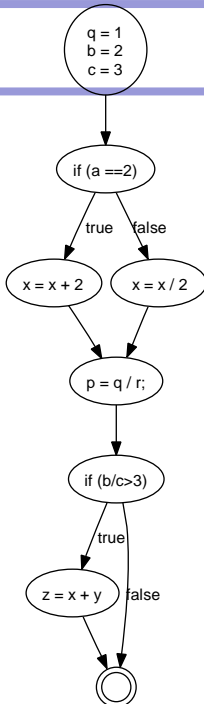
Exercício

Leitura Recomendada

Referências

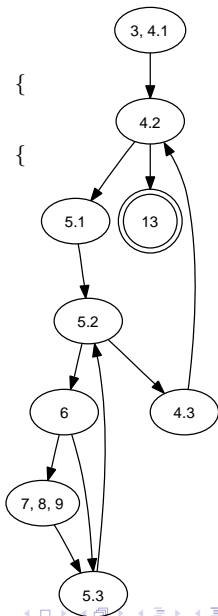
Programa Exemplo (Copeland, 2004)

```
1  q = 1;  
2  b = 2;  
3  c = 3;  
4  if (a ==2) {  
5      x = x + 2;  
6  } else {  
7      x = x / 2;  
8  }  
9  p = q / r;  
10 if (b/c>3) {  
11     z = x + y;  
12 }
```



Programa Sort – Bolha

```
2   public void bolha(int [] a, int size) {  
3       int i, j, aux;  
4       for (i = 0; i < size; i++) {  
5           for (j = size - 1; j > i; j--) {  
6               if (a[j - 1] > a[j]) {  
7                   aux = a[j - 1];  
8                   a[j - 1] = a[j];  
9                   a[j] = aux;  
10            }  
11        }  
12    }  
13 }
```



Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclomática

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

Exercício (1)

Gerar os GFCs para os métodos a seguir:

```
15 void insercao(int a[], int size) {  
16     int i, j, aux;  
17     for (i = 1; i < size; i++) {  
18         aux = a[i];  
19         j = i - 1;  
20         while (j >= 0 && a[j] >= aux) {  
21             a[j + 1] = a[j];  
22             j--;  
23         }  
24         a[j + 1] = aux;  
25     }  
26 }
```

Código fonte completo: [clique aqui](#).

Exercício (2)

```
28 public static void heapsort(int n, double ra[]) {
29     int l, j, ir, i;
30     double rra;
31
32     l = (n >> 1) + 1;
33     ir = n;
34     for (;;) {
35         if (l > 1) {
36             rra = ra[--l];
37         } else {
38             rra = ra[ir];
39             ra[ir] = ra[1];
40             if (--ir == 1) {
41                 ra[1] = rra;
42                 return;
43             }
44         }
45         i = l;
46         j = l << 1;
47         while (j <= ir) {
48             if (j < ir && ra[j] < ra[j + 1]) {
49                 ++j;
50             }
51             if (rra < ra[j]) {
52                 ra[i] = ra[j];
53                 j += (i = j);
54             } else {
55                 j = ir + 1;
56             }
57         }
58         ra[i] = rra;
59     }
60 }
```


Exercício (3)

```
62 void quicksort(int a[], int lo0, int hi0) {
63     int lo = lo0;
64     int hi = hi0;
65     int mid;
66
67     // pause for redraw
68     if (hi0 > lo0) {
69         mid = a[(lo0 + hi0) / 2];
70
71         while (lo <= hi) {
72             while ((lo < hi0) && (a[lo] < mid))
73                 ++lo;
74
75             while ((hi > lo0) && (a[hi] > mid))
76                 --hi;
77
78             if (lo <= hi) {
79                 swap(a, lo, hi);
80                 ++lo;
81                 --hi;
82             }
83         }
84
85         if (lo0 < hi)
86             quicksort(a, lo0, hi);
87
88         if (lo < hi0)
89             quicksort(a, lo, hi0);
90     }
91 }
92
```

Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

 Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclômática

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

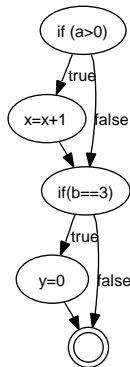
Referências

Níveis de Cobertura (1)

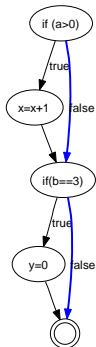
- ▶ Diferentes níveis de cobertura podem ser definidos em função dos elementos do GFC.
- ▶ Cobertura: porcentagem dos requisitos que foram testados *versus* o total de requisitos gerados.
- ▶ Oito diferentes níveis de cobertura são definidos por Copeland (2004).
- ▶ Quanto maior o nível, maior o rigor do critério de teste, ou seja, mais caso de teste ele exige para ser satisfeito.
 - ▶ Nível 0 \leftarrow Nível 1 \leftarrow Nível 2 \leftarrow Nível 3 \leftarrow Nível 4 \leftarrow Nível 5 \leftarrow Nível 6 \leftarrow Nível 7

- ▶ Nível 0: qualquer valor de cobertura inferior a 100% da cobertura de todos os comandos.
- ▶ Nível 1: 100% de cobertura de comandos.
 - ▶ Também chamado de cobertura de nós (critério **todos-nós**)

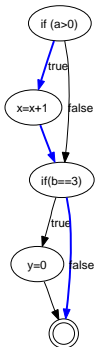
```
1  if (a>0){  
2      x=x+1;  
3  }  
4  if (b==3){  
5      y=0;  
6  }
```



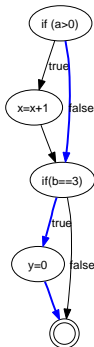
Nível 1: 100% de cobertura de comandos (requisito mínimo de teste)



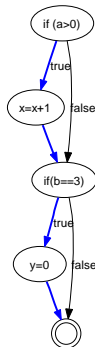
Caminho 1



Caminho 2



Caminho 3



Caminho 4

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".

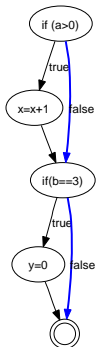
Nível 1: 100% de cobertura de comandos (requisito mínimo de teste)

- ▶ Embora seja o nível mais baixo de cobertura pode ser difícil de ser atingida em alguns casos.
 - ▶ Código para situações excepcionais: falta de memória, disco cheio, arquivos ilegíveis, perda de conexão, dentre outras.
 - ▶ Pode ser difícil ou impossível simular tais situações excepcionais.
 - ▶ Nessas situações, o código correspondente permanece não testado.

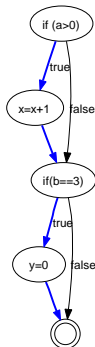
Níveis de Cobertura (2)

- ▶ Nível 2: 100% de cobertura de decisões.
 - ▶ Também chamado de cobertura de arcos/arestas (critério **todos-arcos**).
 - ▶ Objetivo fazer cada comando de decisão assumir os valores TRUE e FALSE.

Nível 2: 100% de cobertura de decisões.



Caminho 1



Caminho 4

Dois casos de testes são suficientes para cobrir todos os arcos do GFC. Por exemplo, $a=2, b=2$ e $a=4, b=3$ satisfazem o critério **todos-arcos**.

- ▶ Nível 3: 100% de cobertura de condições.
 - ▶ Nem todos comandos de decisão são simples como o anterior.
 - ▶ Considere o exemplo abaixo:

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

- ▶ Comando linha 2: requer que $a>0$ e $c==1$ sejam ambos TRUE.
 - ▶ Se a for 0, o comando $c==1$ pode nunca ser executado (linguagem de programação – curto-circuito).
- ▶ Comando linha 5: requer que $b==3$ ou $d<0$ seja TRUE.

- ▶ Nível 3: 100% de cobertura de condições.

- ▶ Para o exemplo em questão:

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

- ▶ Dois casos de teste necessários para cobrir todas as condições:
 $\{a > 0, c = 1, b = 3, d < 0\}$ e $\{a \leq 0, c \neq 1, b \neq 3, d \geq 0\}$
 - ▶ Cobertura de condição é, em geral, melhor que cobertura de decisão: cada condição individual assume os valores TRUE e FALSE.

- ▶ Nível 4: 100% de cobertura decisões/condições.

- ▶ Considere o exemplo abaixo:

```
1  if (x && y) {  
2      conditionedStatement ;  
3  }
```

- ▶ Cobertura de condições pode ser obtida com dois casos de testes:

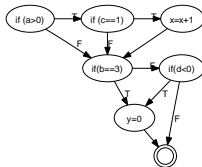
{x=TRUE, y=FALSE } e {x=FALSE, y=TRUE }

mas o comando `conditionedStatement` não será executado.

- ▶ O critério 100% de cobertura de decisões/condições requer que todas as combinações sejam testadas.

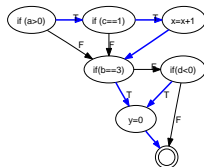
- ▶ Nível 5: 100% de cobertura de condições múltiplas.
 - ▶ Consiste em utilizar o conhecimento de como o compilador avalia condições múltiplas de determinado comando de decisão e utilizar essa informação na geração de casos de testes.
 - ▶ Considere um compilador que avalia condições múltiplas em uma decisão conforme ilustrado abaixo:

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



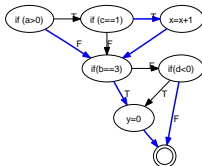
Obter 100% cobertura de condições múltiplas implica cobrir 100% dos critérios anteriores, mas não garante cobertura de todos-caminhos.

- ▶ Nível 5: 100% de cobertura de condições múltiplas.
 - ▶ Cobrir os arcos do grafo abaixo requer os seguintes casos de testes:



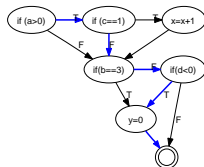
a>0 c=1 b=3 d<0 ←

- Nível 5: 100% de cobertura de condições múltiplas.
 - Cobrir os arcos do grafo abaixo requer os seguintes casos de testes:



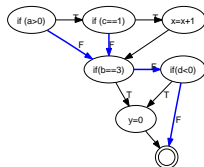
$a > 0$	$c = 1$	$b = 3$	$d < 0$
$a \leq 0$	$c = 1$	$b = 3$	$d \geq 0 \leftarrow$

- ▶ Nível 5: 100% de cobertura de condições múltiplas.
 - ▶ Cobrir os arcos do grafo abaixo requer os seguintes casos de testes:



$a > 0$	$c = 1$	$b = 3$	$d < 0$
$a \leq 0$	$c = 1$	$b = 3$	$d \geq 0$
$a > 0$	$c \neq 1$	$b \neq 3$	$d < 0 \leftarrow$

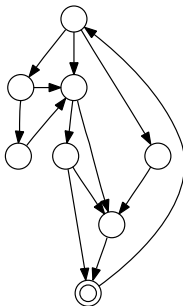
- Nível 5: 100% de cobertura de condições múltiplas.
 - Cobrir os arcos do grafo abaixo requer os seguintes casos de testes:



$a > 0$	$c = 1$	$b = 3$	$d < 0$
$a \leq 0$	$c = 1$	$b = 3$	$d \geq 0$
$a > 0$	$c \neq 1$	$b \neq 3$	$d < 0$
$a \leq 0$	$c \neq 1$	$b \neq 3$	$d \geq 0 \leftarrow$

- ▶ Nível 6: cobertura de loop.
 - ▶ Quando programas possuem loop, o número de caminhos possíveis pode ser infinito.
 - ▶ O número de caminhos pode ser reduzido limitando a execução do loop a:
 - ▶ 0 vezes.
 - ▶ 1 vez.
 - ▶ 2 vezes.
 - ▶ n vezes (sendo n um número de vezes padrão que o loop é executado).
 - ▶ m vezes (sendo m o número máximo de vezes que o loop pode ser executado).
 - ▶ $m - 1$ vezes.
 - ▶ $m + 1$ vezes.

- ▶ Nível 7: 100% de cobertura de caminhos.
 - ▶ Também conhecido como critério **todos-caminhos**.
 - ▶ Para programas sem loop o número de caminhos pode ser pequeno o suficiente e casos de testes podem ser construídos para cobri-los.
 - ▶ Para programas com loop o número de caminhos pode ser muito grande ou infinito, tornando-se impossível cobrir todos os caminhos de execução.



Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclométrica

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

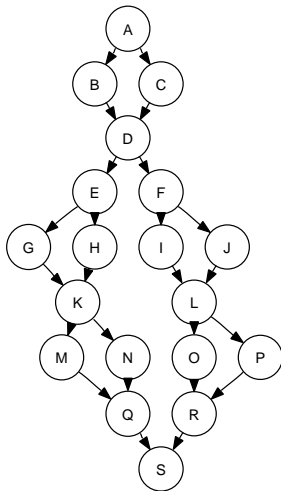
Teste do Caminho Básico

- ▶ Critério de teste estrutural baseado no fluxo de controle.
- ▶ Trabalho pioneiro desenvolvido por McCabe (1976).
- ▶ Baseado no conceito de Complexidade Ciclomática (calculada a partir do GFC).

Passos para Aplicação

- ▶ Construir o GFC para o módulo do produto em teste.
- ▶ Calcular a Complexidade Ciclomática (\mathcal{C}).
- ▶ Selecionar um conjunto de \mathcal{C} caminhos básicos.
- ▶ Criar um caso de teste para cada caminho básico.
- ▶ Executar os casos de testes.

Cálculo da Complexidade Ciclomática



- ▶ Considere o GFC ao lado.
- ▶ McCabe (1976) define a Complexidade Ciclomática (C) como:

$$C = \text{arcos} - \text{nós} + 2$$

$$C = 24 - 19 + 2$$

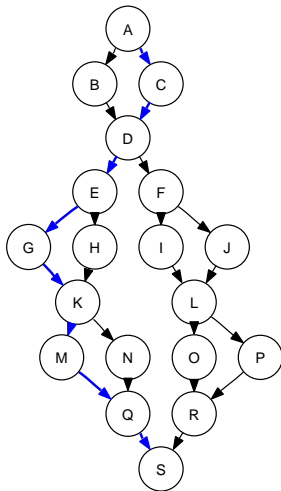
$$C = 7$$

Para um GFC com p decisões binárias (dois arcos saindo):

$$C = p + 1$$

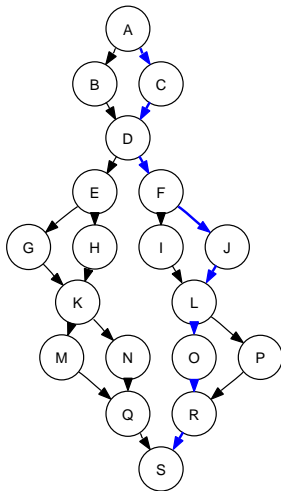
- ▶ A complexidade ciclomática representa o número mínimo de caminhos independentes, sem loop, que gera todos os possíveis caminhos de um módulo.
- ▶ Em termos do GFC, cada caminho básico inclui pelo menos um arco que ainda não foi selecionado.
- ▶ Criar e executar C casos de testes (um para cada caminho básico) garante cobertura dos critérios **todos-nós** e **todos-arcos**.

Criação do Conjunto de Caminhos Básicos - Passo 2



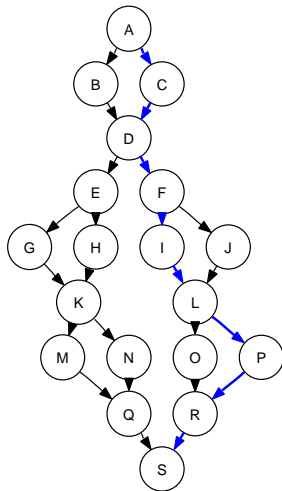
- ▶ Altere a saída do primeiro comando de decisão e mantenha o máximo possível do caminho inalterado.
- ▶ Caminho 2: ACDEGKM QS

Criação do Conjunto de Caminhos Básicos - Passo 5 (1)



- ▶ Todas as decisões do caminho básico foram contempladas.
- ▶ A partir do segundo caminho, fazer as inversões dos comandos de decisão até o final do GFC.
- ▶ Esse padrão é seguido até que o conjunto completo de caminhos seja atingido.
- ▶ Caminho 6: ACDFJLORS

Criação do Conjunto de Caminhos Básicos - Passo 5 (2)



► Continuação do passo anterior.

► Caminho 7: ACDFILPRS

Exemplo de Aplicação do Critério

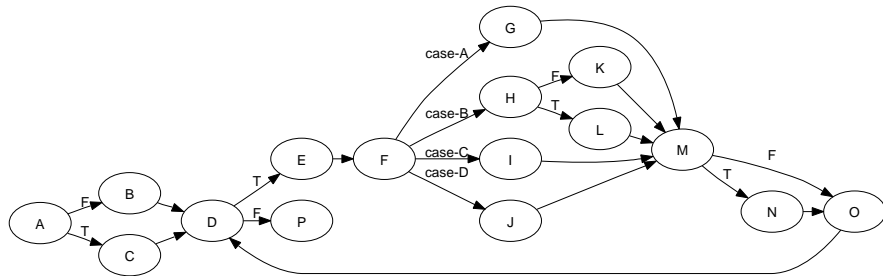
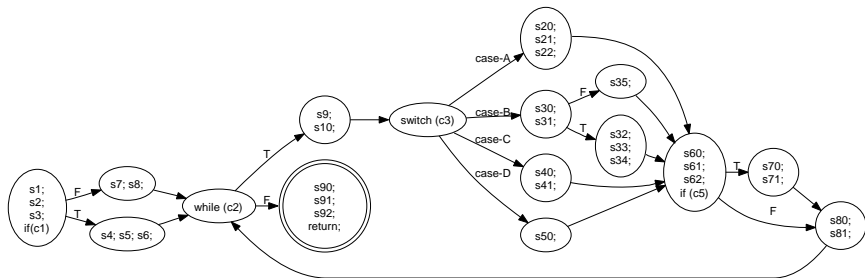
Considere o código Java abaixo:

```
1  boolean evaluateBuySell (TickerSymbol ts) {
2      s1;
3      s2;
4      s3;
5      if (c1) {s4; s5; s6;}
6      else {s7; s8;}
7      while (c2) {
8          s9;
9          s10;
10         switch (c3) {
11             case-A:
12                 s20;
13                 s21;
14                 s22;
15                 break; // End of Case-A
16             case-B:
17                 s30;
18                 s31;
19                 if (c4) {
20                     s32;
21                     s33;
22                     s34;
23                 }
24                 else {
25                     s35;
26                 }
27                 break; // End of Case-B
28             case-C:
29                 s40;
30                 s41;
31                 break; // End of Case-C
32             case-D:
33                 s50;
34                 break; // End of Case-D
35         } // End Switch
36         s60;
37         s61;
38         s62;
39         if (c5) {s70; s71; }
40         s80;
41         s81;
42     } // End While
43     s90;
44     s91;
45     s92;
46     return result;
```

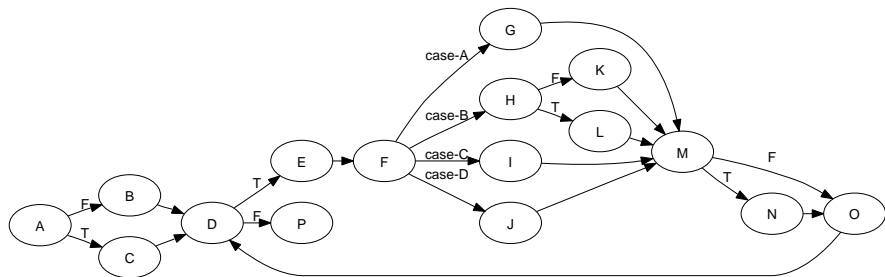
Técnica de Teste Caixa Branca

└ Teste do Caminho Básico

└ Exemplo



Complexidade Ciclomática



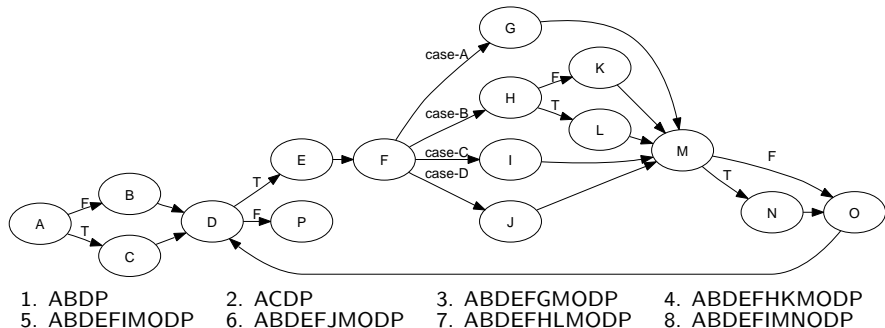
- Cálculo da complexidade ciclomática para o GFC acima:

$$C = \text{arcos} - \text{nós} + 2$$

$$C = 22 - 16 + 2$$

$$C = 8$$

Caminhos Básicos



Conjunto de Teste

- | | | | |
|---------------|---------------|----------------|----------------|
| 1. ABDP | 2. ACDP | 3. ABDEFGMODP | 4. ABDEFHKMODP |
| 5. ABDEFIMODP | 6. ABDEFJMODP | 7. ABDEFHLMODP | 8. ABDEFIMNODP |

Caso Teste	C1	C2	C3	C4	C5
1	False	False	N/A	N/A	N/A
2	True	False	N/A	N/A	N/A
3	False	True	A	N/A	False
4	False	True	B	False	False
5	False	True	C	N/A	False
6	False	True	D	N/A	False
7	False	True	B	True	False
8	False	True	C	N/A	True

Aplicabilidade e Limitações

- ▶ Critérios de Fluxo de Controle são a “pedra fundamental” do teste de unidade.
- ▶ Devem ser aplicados a todos os módulos do software, em especial, nos mais críticos.
- ▶ Exigem habilidades de programação do testador para compreender o fluxo de controle do programa.
- ▶ Pode consumir tempo e recursos significativos para sua aplicação.

Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclomática

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

Resumo

- ▶ Critérios de Fluxo de Controle identificam caminhos que devem ser percorridos no código do programa.
- ▶ GFC é a base a partir do qual os requisitos de testes são derivados.
- ▶ Complexidade Ciclomática define o número mínimo de conjuntos de caminhos independentes livres de loop (caminho básico).
- ▶ O conjunto de caminhos básicos inclui todos os nós e arcos do GFC.
- ▶ Casos de testes que exercitem todos os caminhos básicos do conjunto também executam todos os comandos e todas as decisões do programa.

Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclômática

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

Exercício

- Considere o programa abaixo: 1) crie o GFC; 2) determine a complexidade ciclomática; 3) derive um conjunto de caminhos básicos; e 4) desenvolva os casos de testes necessários para executá-los.

```
1  if (c1) {  
2      while (c2) {  
3          if (c3) {  
4              s1; s2;  
5              if (c4) s3;  
6              else s4;  
7              break; // Skip to end of while  
8          }  
9      }  
10     else  
11         if (c5) { s5; }  
12         else {  
13             s6;  
14             s7;  
15             break;  
16         }  
17 } // End of while  
18 } // End of if  
19 s8;  
20 if (c6)  
21     s9;  
22     s10;  
23     s11;
```


Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclômática

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

Leitura Recomendada

Mais informações sobre esse tema podem ser encontrados em:

- ▶ Seção 2, Capítulo 10 do livro de Copeland (2004).

Critérios Baseados em Fluxo de Controle

Definições

 Grafo de Fluxo de Controle

Exemplo de GFC

Exercícios

Níveis de Cobertura

Teste do Caminho Básico

 Passos do Teste de Caminho Básico

 Complexidade Ciclomática

 Criação do Conjunto de Caminhos

 Exemplo

 Aplicabilidade e Limitações

Resumo

Exercício

Leitura Recomendada

Referências

Referências Bibliográficas

Copeland, L. *A practitioner's guide to software test design*.
Artech House Publishers, 2004.

McCabe, T. A complexity measure. *IEEE Transactions on Software Engineering*, v. 2, n. 4, p. 308–320, 1976.