

SSC721 – Teste e Inspeção de Software

Técnica de Teste Estrutural

Profa. Ellen Francine Barbosa
francine@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação — ICMC/USP

- Técnica Estrutural
- Critérios da Técnica Estrutural
- Critérios de Fluxo de Controle
- Critério Baseado na Complexidade
- Resumo
- Exercícios

- Conhecida como teste caixa-branca (em oposição ao teste caixa-preta).
- Baseia-se no conhecimento da **estrutura interna** (implementação) do programa.
 - Teste dos detalhes procedimentais.
- A maioria dos critérios dessa técnica utiliza uma representação de programa conhecida como **grafo de programa**.

- Passos básicos para aplicar um critério de teste estrutural:
 - A implementação do produto em teste é analisada.
 - **Caminhos** através da implementação são escolhidos.
 - Valores de entrada são selecionados de modo que os caminhos escolhidos sejam executados.
 - As saídas esperadas para as entradas selecionadas são determinadas.
 - Os casos de testes são construídos.
 - As saídas obtidas são comparadas às saídas esperadas.
 - Um relatório é gerado para avaliar o resultado dos testes.

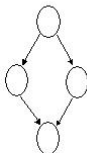
- Representação do programa que visa auxiliar a geração dos requisitos de teste.
- Seja $GFC = (N, E)$ um grafo no qual **N** representa o conjunto de nós e **E** o conjunto de arcos.
 - **Nós**: blocos de comandos **indivisíveis**.
 - Não existe desvio para o meio do bloco.
 - Uma vez que o primeiro comando do bloco é executado, os demais comandos são executados sequencialmente.
 - **Arestas ou Arcos**: representam o fluxo de controle entre os nós.
 - **Caminhos**: seqüências de execução de comandos que iniciam em um nó de entrada e terminam em um nó de saída.

• Construções Básicas de um GFC:

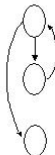
seqüência



if



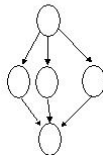
while



repeat



case



O programa *Identifier* determina se um identificador é válido ou não. Um identificador válido deve começar com uma letra e conter apenas letras ou dígitos. Além disso, deve ter no mínimo um caractere e no máximo seis caracteres de comprimento.

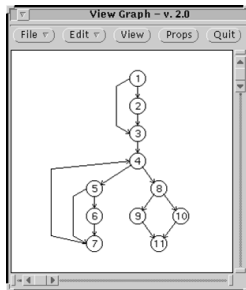
```

/* 01 */      {
/* 01 */      char  achar;
/* 01 */      int length, valid_id;
/* 01 */      length = 0;
/* 01 */      printf ("Identificador: ");
/* 01 */      achar = fgetc (stdin);
/* 01 */      valid_id = valid_s(achar);
/* 01 */      if (valid_id)
/* 02 */          length = 1;
/* 03 */      achar = fgetc (stdin);
/* 04 */      while (achar != '\n')
/* 05 */      {
/* 05 */          if (!(valid_f(achar)))
/* 06 */              valid_id = 0;
/* 07 */          length++;
/* 07 */          achar = fgetc (stdin);
/* 07 */      }
/* 08 */      if (valid_id && (length >= 1) && (length < 6))
/* 09 */          printf ("Valido\n");
/* 10 */      else
/* 10 */          printf ("Invalido\n");
/* 11 */      }

```

Implementação do Programa *Identifier* (função main).

- Função `valid_s()`: determina se o primeiro caractere é válido.
- Função `valid_f()`: determina se o próximo caractere é válido.



Grafo de Programa do *Identifier* Gerado pela *View-Graph*

- Nós: 1, 2, 3, ...
- Arcos: $\langle 1,2 \rangle$, $\langle 1,3 \rangle$, ...
- Arcos Primitivos:
 $\langle 1,2 \rangle$, $\langle 1,3 \rangle$, $\langle 5,6 \rangle$, $\langle 5,7 \rangle$, $\langle 8,9 \rangle$, $\langle 8,10 \rangle$
- Caminhos
 - Completo: (1,2,3,4,5,7,4,8,9,11)
 - Simples: (1,2,3,4,5,6,7), (4,5,6,7,4)
 - Livre de Laço: (1,2,3,4,5,7)

- Critérios da técnica estrutural podem ser utilizados em todas as fases de teste.
- Em geral, bastante aplicado no teste de unidade pelo próprio desenvolvedor da mesma.
 - Garantir que a lógica da unidade em teste está correta.
- Teste de caminhos:
 - Caminhos dentro de uma unidade.
 - Caminhos entre unidades.
 - Caminhos entre sub-sistemas.
 - Caminhos entre o sistema todo.

- O número de **caminhos** a serem executados pode ser infinito (semelhante ao teste exaustivo).
- Assume o fluxo de controle **correto** (ou próximo do correto).
 - Casos de testes são baseados em caminhos existentes: caminhos inexistentes não podem ser descobertos.
 - Caminhos ausentes: se o programa não implementa algumas funções, não existirá um caminho que corresponda àquela função; conseqüentemente, nenhum dado de teste será requerido para exercitá-la.

- 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.
- Implementação dos critérios de teste é mais fácil de ser **automatizada**.
 - Ferramentas de teste.

- Baseados em Fluxo de Controle
 - Todos-Nós
 - Todas-Arestas
 - Todos-Caminhos: Simples, Completo, Livre de Laço, ...
- Baseados em Complexidade
 - Critério de McCabe (teste do caminho base).
- Baseados em Fluxo de Dados
 - Critérios de Rapps & Weyuker
 - Todas-Defs, Todos-Usos, Todos-P-Usos e outros.
 - Critérios Potenciais-Usos
 - Todos-Potenciais-Usos, Todos-Potenciais-Usos/DU e outros.

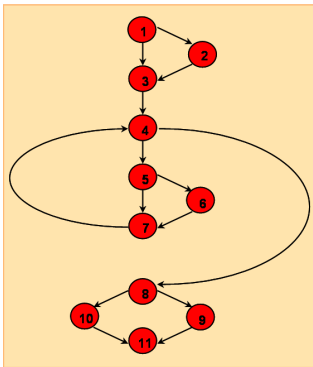
- Utilizam características de controle da execução do programa para determinar os **requisitos de teste**.
 - Comandos
 - Desvios de Execução
- Alguns critérios de fluxo de controle:
 - Todos-Nós
 - Todos-Arcos
 - Todos-Caminhos
 - Todos-Caminhos-Simples
 - Todos-Caminhos-Completo
 - Todos-Caminhos-Livre-Laço

- Exige que a execução do programa passe, ao menos uma vez, em cada **nó** do grafo de programa.

Ou seja...

Requer que cada **comando** do programa seja executado pelo menos uma vez.

- Elementos Requeridos:
 - Nós: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11



Critério Todos-Nós.

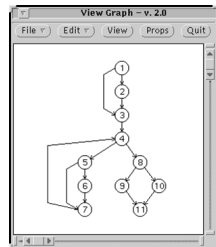
- Exige que a execução do programa passe, ao menos uma vez, em cada **arco** do grafo de programa.

Ou seja...

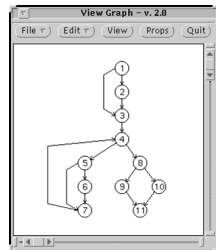
Requer que cada **desvio de fluxo de controle** do programa seja exercitado pelo menos uma vez.



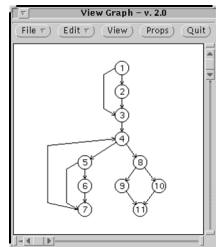
- **Todos-Caminhos-Completo**: requer que todos os **caminhos completos** sejam exercitados pelo menos uma vez.
 - Um **caminho completo** é um caminho P em que o primeiro nó de P é o **nó inicial** e o último nó de P é o **nó final** do grafo.
- Exemplo de elementos requeridos (programa identifier):
 - (1,2,3,4,8,10,11)
 - (1,2,3,4,5,7,4,5,7,4,8,10,11)
 - ...



- **Todos-Caminhos-Simples**: requer que todos os **caminhos simples** sejam exercitados pelo menos uma vez.
 - Um **caminho simples** é formado por nós **distintos**, exceto possivelmente o primeiro e o último.
- Exemplo de elementos requeridos (programa identifier):
 - (1,2,3,4,8,10,11)
 - (4,5,6,7,4)
 - ...

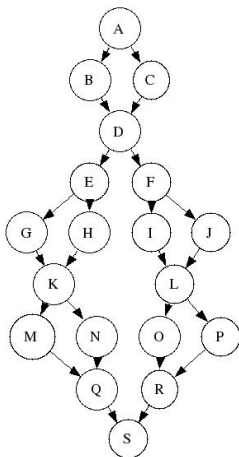


- **Todos-Caminhos-Livre-Laços**: requer que todos os **caminhos livre de laço** sejam exercitados pelo menos uma vez.
 - Um **caminho livre de laço** é um caminho simples em que **todos os nós são distintos**, inclusive o primeiro e o último.
- Exemplo de elementos requeridos (programa identifier):
 - (1,2,3,4,8,10,11)
 - (1,2,3,4,5,7)
 - ...



- Conhecido como **teste do caminho básico** ou critério de McCabe
- Baseado no conceito de **Complexidade Ciclométrica**, calculada a partir do GFC.
- A complexidade ciclométrica define o número de **caminhos independentes** de um programa.
 - Um caminho independente é qualquer caminho que introduz no mínimo uma **nova aresta** ainda não atravessada pelos caminhos anteriores.
 - Limite **máximo** para o número de caminhos independentes que constitui o conjunto básico e, conseqüentemente, um limite máximo do número de casos de teste que deve ser projetado e executado para garantir a cobertura de todas as instruções de programa.

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



- Considere o GFC ao lado.
- McCabe 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):

$$C = p + 1$$

- Cada caminho básico percorre **pelo menos** um arco que ainda não foi percorrido.

Criação do Conjunto de Caminhos Básicos: Passo 1

SSC721 – Teste e
Inspeção de Software

Técnica Estrutural

Crítérios da Técnica
Estrutural

Crítérios de Fluxo de
Controle

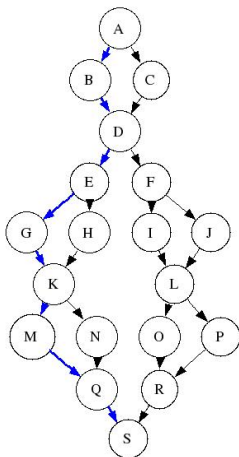
Crítério Baseado na
Complexidade

Cálculo da Complexidade
Ciclomática

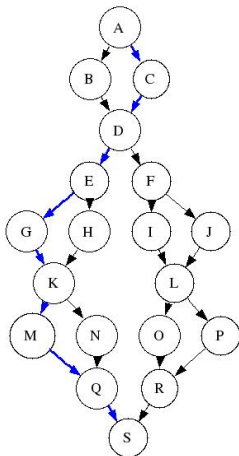
Criação do Conjunto de Caminhos
Básicos

Resumo

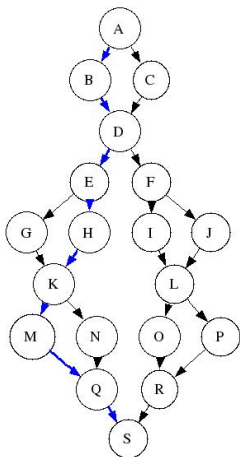
Exercícios



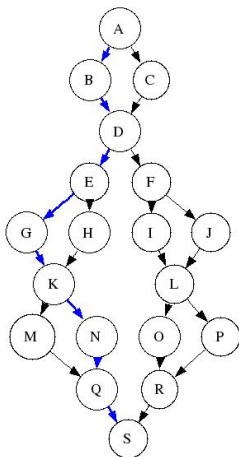
- Escolha um caminho básico.
Esse caminho pode ser:
 - Caminho mais comum.
 - Caminho mais crítico.
 - Caminho mais importante do ponto de vista de teste.
- Caminho 1: **ABDEGKMQS**



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



- A partir do caminho básico alterar a saída do terceiro comando de decisão. Repetir esse processo até atingir o final do GFC.
- Caminho 4: **ABDEHKMQS**



- Continuação do passo anterior.

- Caminho 5: **ABDEGKNQS**

Criação do Conjunto de Caminhos Básicos: Passo 6

SSC721 – Teste e
Inspeção de Software

Técnica Estrutural

Crítérios da Técnica
Estrutural

Crítérios de Fluxo de
Controle

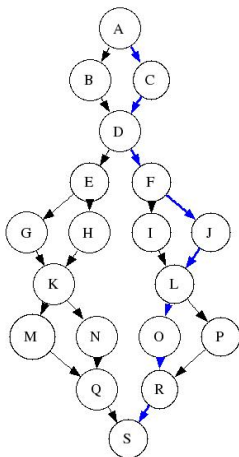
Crítério Baseado na
Complexidade

Cálculo da Complexidade
Ciclomática

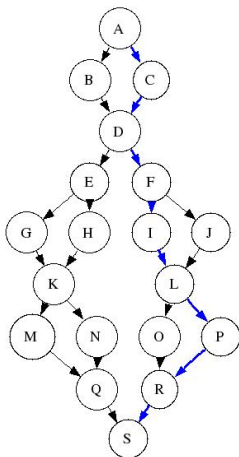
Criação do Conjunto de Caminhos
Básicos

Resumo

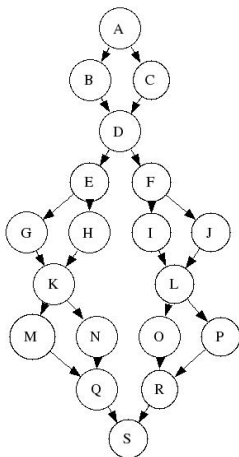
Exercícios



- 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**

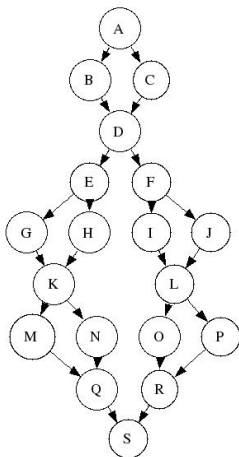


- Continuação do passo anterior.
- Caminho 7: **ACDFILPRS**



- Requisitos de testes derivado pelo critério:

- ABDEGKMQS
- ACDEGKMQS
- ABDFILORS
- ABDEHKMQS
- ABDEGKNQS
- ACDFJLORS
- ACDFILPRS



- Conjunto criado **não é único**.
- **Propriedade**: o conjunto de teste que exercita os caminhos básicos também exercita todos os nós e todos os arcos do programa.
 - Garante a cobertura dos critérios **todos-nós** e **todos-arcos**.

- Critérios da técnica estrutural identificam **caminhos** que devem ser percorridos no **código** do programa.
- **GFC** é a base a partir do qual os requisitos de testes são derivados.

- Podem ser aplicados em todos os módulos do software, em especial, nos mais **críticos**.
- Exigem **habilidades de programação** do testador para compreender o fluxo de controle do programa.
- Podem consumir **tempo** e **recursos** significativos para sua aplicação.



Resumindo... Critérios Baseados na Complexidade

SSC721 – Teste e
Inspeção de Software

Técnica Estrutural

Critérios da Técnica
Estrutural

Critérios de Fluxo de
Controle

Critério Baseado na
Complexidade

Resumo

Exercícios

- 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.

SSC721 – Teste e
Inspeção de Software

Técnica Estrutural

Critérios da Técnica
Estrutural

Critérios de Fluxo de
Controle

Critério Baseado na
Complexidade

Resumo

Exercícios

Exercício: Fluxo de Controle



- Gere **casos de teste** para executar **todos os nós e todos os arcos** do programa identifier. Identifique o **caminho** percorrido para cada caso de teste gerado.

```

/* 01 */      {
/* 01 */      char  achar;
/* 01 */      int length, valid_id;
/* 01 */      length = 0;
/* 01 */      printf ("Identificador: ");
/* 01 */      achar = fgetc (stdin);
/* 01 */      valid_id = valid_s(achar);
/* 01 */      if (valid_id)
/* 02 */          length = 1;
/* 03 */      achar = fgetc (stdin);
/* 04 */      while (achar != '\n')
/* 05 */      {
/* 05 */          if (!(valid_f(achar)))
/* 06 */              valid_id = 0;
/* 07 */          length++;
/* 07 */          achar = fgetc (stdin);
/* 07 */      }
/* 08 */      if (valid_id && (length >= 1) && (length < 6))
/* 09 */          printf ("Valido\n");
/* 10 */      else
/* 10 */          printf ("Invalido\n");
/* 11 */      }

```