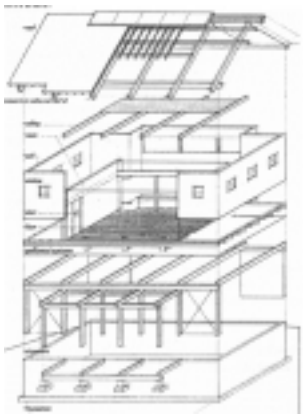
PUC CAMPINAS

Engenharia de Software

Tema da Aula

Teste de Software
II – Técnicas de Teste



Prof. Cristiano R R Portella
portella@widesoft.com.br

PUC CAMPINAS
Engenharia de Software

Ciclo de Vida do Software e a Atividade de Teste

Ambiente de Desenvolvimento		**Ambiente de Produção**	
Planejamento	Análise	Utilização	Gerência e Controle
Projeto	Codificação		
VV&T: Verificação, Validação e Testes (Unitário, Integração e Aceitação)		Teste de performance, recuperação Simulação de Carga, Segurança etc.	
Ambiente de Teste		**Ambiente de Simulação**	

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste
Tipos Básicos

Teste Funcional (ou Caixa Preta)

Este método testa o software a partir de sua funcionalidade (comportamento), através de suas entradas e saídas (vê o produto como uma caixa preta; não considera os detalhes de implementação, ou seja: não observa o código interno).

O Conjunto de casos de teste funcional deve exercitar todos os requisitos do software e o testador deriva casos de teste a partir da especificação do software.

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste
Tipos Básicos

Teste Funcional (ou Caixa Preta)

Os testes são realizados para demonstrar que:

1. As funções são operacionais e isentas de falhas;
2. As entradas são adequadas e as saídas são corretamente produzidas; e
3. A integridade das informações processadas é mantida.

Estes testes são realizados por desenvolvedores e usuários (isolada e/ou conjuntamente).

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste
Tipos Básicos

Teste Estrutural (ou Caixa Branca)

Baseia-se num minucioso exame dos detalhes da codificação (algoritmo).

Método, através do qual, o testador analisa o código gerado (algoritmo), derivando casos de teste a partir da estrutura interna do software.

Estes testes são executados por desenvolvedores do software (não necessariamente o(s) programadores do código em teste).

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste
Tipos Básicos

Teste Estrutural (ou Caixa Branca)

O Conjunto de casos de teste estrutural provê medidas para avaliar o grau de cobertura (% de cobertura; pelo menos uma execução) de elementos da estrutura do software (**módulos**, **comandos**, etc).

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste

1- Caixa Branca ou Estrutural ou Teste em pequeno porte:

1.1 Teste de Caminho Básico (estruturas de controle)
1.2 Teste de Condição (condições lógicas)
1.3 Teste de Fluxo de Dados
1.4 Teste de Laços
. . .

2- Caixa Preta ou Funcional:

2.1 Particionamento de Equivalência
2.2 Teste de Valor Limite
2.3 Teste de Comparação
. . .

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste

3. Teste de Validação (do tipo caixa preta):

3.1 Testes Alfa e Beta
3.2 Teste de Sistema
3.3 Teste de Recuperação
3.4 Teste de Segurança
3.5 Teste de Estresse
3.6 Teste de Desempenho
3.8 Teste de Cenários
. . .

4. Técnica de "Error Guessing" ou "smelling errors".

PUC CAMPINAS
Engenharia de Software

TESTE ESTRUTURAL

(ou CAIXA BRANCA ou
TESTE EM PEQUENO PORTE)

PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)

Objetivo: Caracterizar que um determinado conjunto de componente elementares (comandos, ramos condicionais, caminhos) foram exercitados (cobertos) por um conjunto de casos de teste.

✓ Útil para as seguintes categorias de defeitos:

- Expressões lógicas e asserções incorretas;
- Erros tipográficos (trocar > por um <);
- Seleção Incorreta de Caminhos; e
- Em casos especiais ou raros (exceções);

PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)

- ✓ Exige instrumentação do programa fonte
- ✓ Técnica Mais Utilizada:

  Teste do Caminho Básico (ou estruturas de controle). Para isso deve-se traçar o Grafo de Fluxo de Controle (GFC), que descreve o fluxo de controle lógico do programa, usando notação própria (transformar um programa em um GFC).

PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)
Grafo de Fluxo de Controle

GFC: É um Grafo Dirigido (dígrafo), com um único nó de entrada e um único nó de saída.

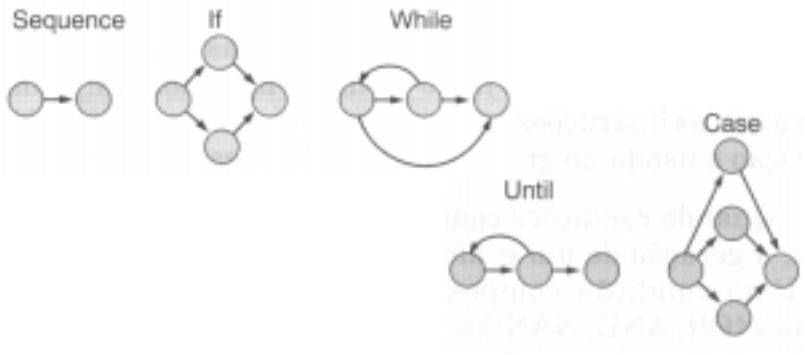
Para obtê-lo é preciso decompor um programa em blocos de comandos, ligando-os através de arcos.

- ✓ **Nós**: Blocos de Comandos
- ✓ **Arcos ou Ramos**: Transferência de fluxo entre blocos
- ✓ **Caminho**: Seqüência de nós $n_1,n_2,n_3,...,n_m$, desde que haja um arco ligando n_i a n_{i+1}.
- ✓ **Região**: Áreas delimitadas por ramos e nós.

PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)
Grafo de Fluxo de Controle

O Grafo de Fluxo de Controle é criado pela transposição do fluxo do programa (dispositivos lógicos) em estruturas padrões:

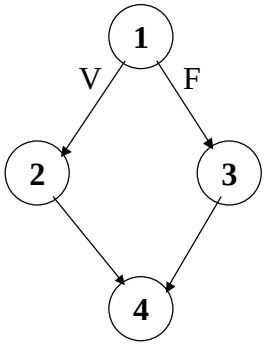


PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)
Grafo de Fluxo de Controle

Notação básica do GFC
If-Then-Else

```
if (condição)  {1}
   bloco 1     {2}
else
    bloco 2;   {3}
end-if         {4}
```

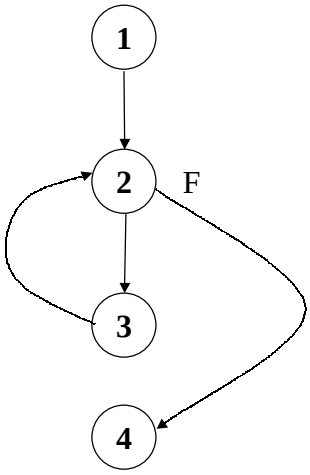


PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)
Grafo de Fluxo de Controle

Notação básica do GFC
Do While

```
while (condição)  {2}
   bloco          {3}
end-while         {4}
```

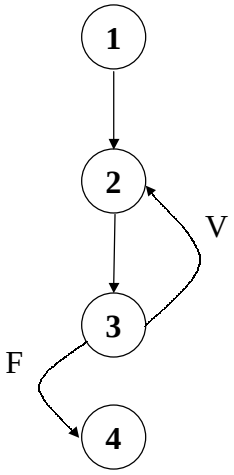


PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)
Grafo de Fluxo de Controle

Notação básica do GFC
Do Until

```
do
   bloco          {2}
until  (condição)  {3}
próximo-comando {4}
```

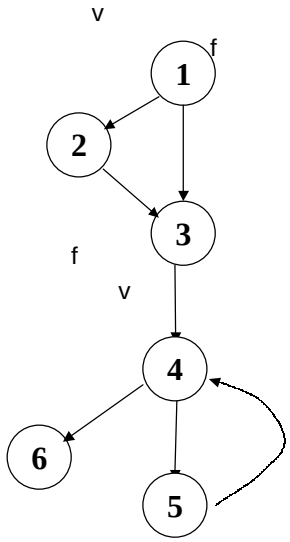


PUC CAMPINAS
Engenharia de Software

Teste Estrutural (Caixa Branca)
Grafo de Fluxo de Controle

Exemplo de GFC

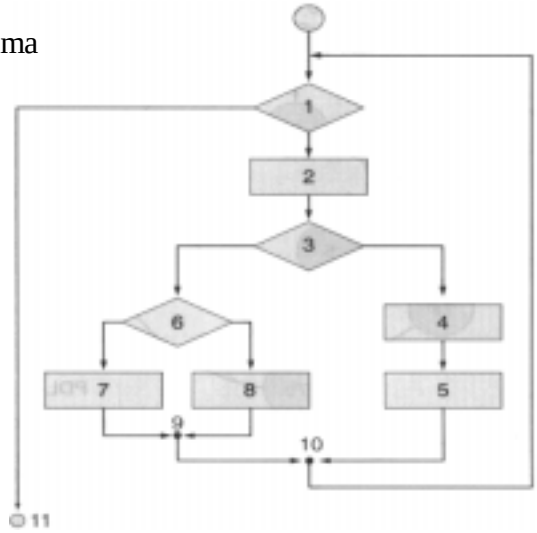
```
Nó Comandos
1 int mdc(int a, int b)
1 { int r,aux;;
1 if (a < b) // inverte a e b
2 { aux = a;
2   a = b;
2   b = aux; }
3 r = a % b;
4 while (r!=0)
5 {   a = b;
5     b = r;
5     r = a % b; }
6  return b;
6 }
```



PUC CAMPINAS
Engenharia de Software

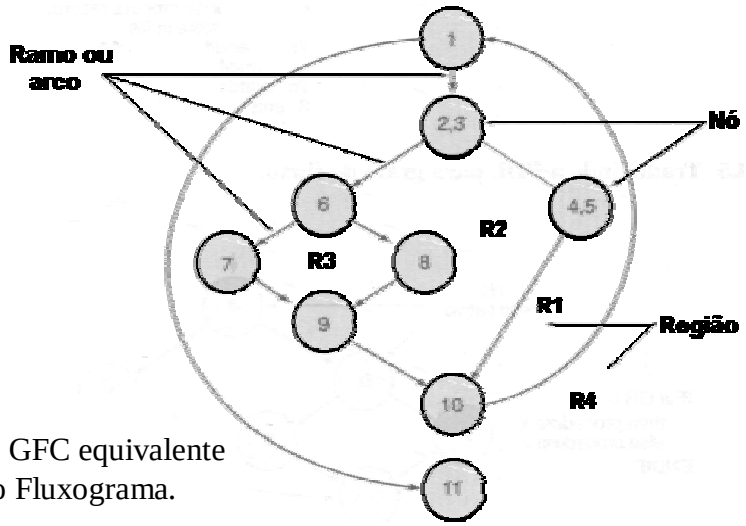
Técnicas de Teste Estrutural
Teste de Caminho Básico

1. Fluxograma



PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Caminho Básico



2. GFC equivalente ao Fluxograma.

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Critérios Estruturais Clássicos

Critérios Estruturais:

Consideram apenas as informações do fluxo de controle do programa.

Critérios Estruturais Clássicos:

- ✓ Todos os Nós
- ✓ Todos os Arcos
- ✓ Todos os Caminhos (Teste Exaustivo)
- ✓ Todos os Caminhos Independentes

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Critérios Estruturais Clássicos

Critério Estrutural: Todos os Nós

- Exige que os casos de teste exercitem todos os nós pelo menos uma vez. Se todos os Nós forem Executados, todos os comandos terão sido exercitados pelo menos uma vez.

Critério Estrutural: Todos os Arcos

- Exige que os casos de teste exercitem todos os arcos pelo menos uma vez. Se todos os Arcos forem Executados, todos os nós terão sido executados pelo menos uma vez.

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Critérios Estruturais Clássicos

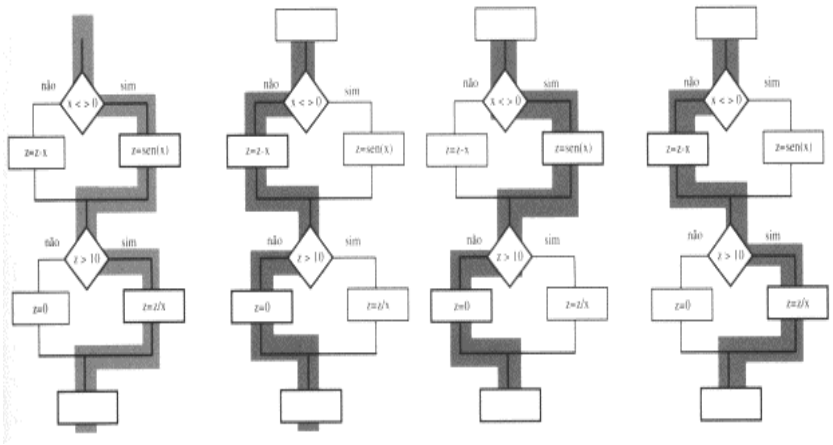
✓ Critério Estrutural: Todos-Caminhos-Independentes

- Baseado na Análise de Complexidade Ciclomática do Código, de McCabe;
- Garante que todos os comandos (nós do GFC) e todos os ramos (arcos do GFC) serão cobertos;
- Caminho Independente é qualquer caminho do programa que introduza pelo menos um novo conjunto de instruções de processamento ou uma nova condição;
- Nó Predicativo: Aquele que tem duas ou mais saídas (condição).

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Todos os Caminhos Independentes

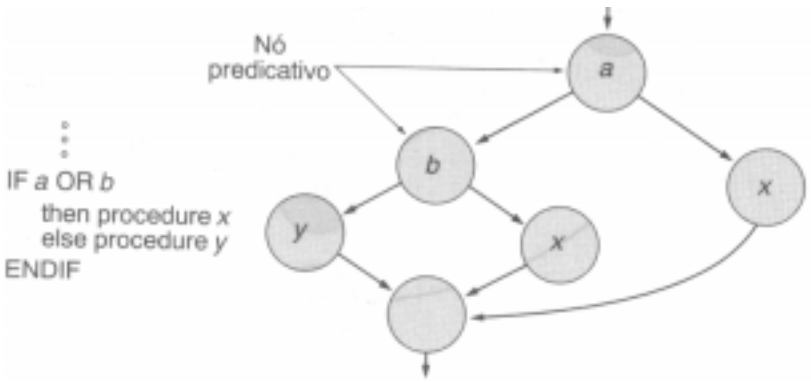
Caminhos Independentes:



PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Critérios Estruturais Clássicos

Nó Predicativo



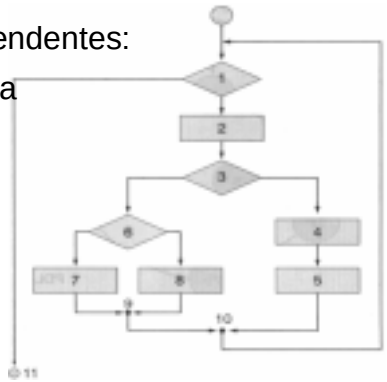
PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Todos os Caminhos Independentes

Exemplo de Caminhos Independentes:

Grafo Derivado do Fluxograma

1. 1-11
2. 1-2-3-4-5-10-1-11
3. 1-2-3-6-8-9-10-1-11
4. 1-2-3-6-7-9-10-1-11



Todos incluem, pelo menos, 1 novo arco

O Caminho 1-2-3-4-5-10-1-2-3-6-8-9-10-1-11 não é independente (é uma simples combinação dos caminhos 2 e 3)

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Todos os Caminhos Independentes

Caminhos Independentes:

a) 1,2,3,4,6

b) 1,3,4,6

c) 1,2,3,4,5,4,6

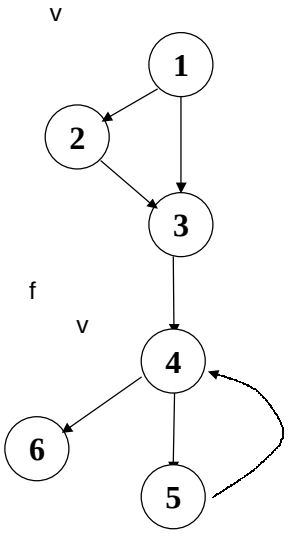
Conjunto Básico não é único.

Outro Conjunto possível:

a) 1,2,3,4,6

b) 1,3,4,5,4,6

- ✓ Aplicação do "Conjunto Máximo" aumenta a confiabilidade do conjunto de casos de teste



PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Todos os Caminhos Independentes

O número de Caminhos Independentes pode ser calculado usando-se o GFC desenhado manualmente ou calculado automaticamente através de programa que calcula a Métrica de complexidade Ciclomática de Thomas McCabe.

A Complexidade Ciclomática V(G) nos oferece um limite máximo para o número de caminhos independentes que constitui o conjunto básico, portanto um limite máximo para o número de testes que deve ser projetado e executado para garantir a cobertura de todas as instruções do programa.

PUC CAMPINAS
Engenharia de Software

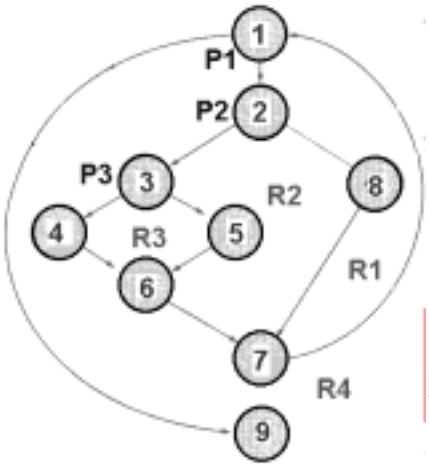
Técnicas de Teste Estrutural
Todos os Caminhos Independentes

Métrica da Complexidade Ciclomática "V(G)" pode ser calculada de três maneiras diferentes:

1. **V(G) = (No_de_Arcos - No-de_Nos) + 2**
2. **V(G) = No_de_Nós_Predicativos + 1**
3. **V(G) = No_de_Regiões**

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Todos os Caminhos Independentes



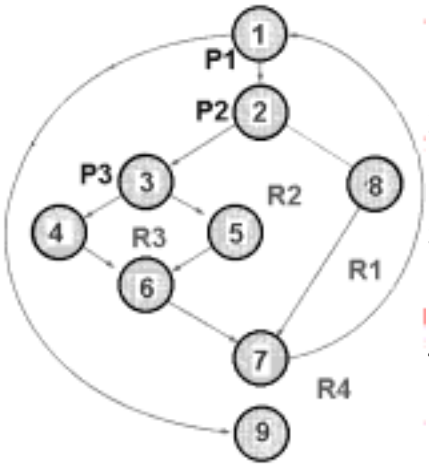
1- V(G) =
(Ramos – Nós) + 2
(11 – 9) + 2
4

2. V(G) = Nós-Predic + 1
3 + 1
4

3. V(G) = No-Regiões
4

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Todos os Caminhos Independentes



1o:
1–9

2o:
1–2–8–7–1–9

3o:
1–2–3–5–6-7–1–9

4o:
1–2–3–4–6-7–1–9

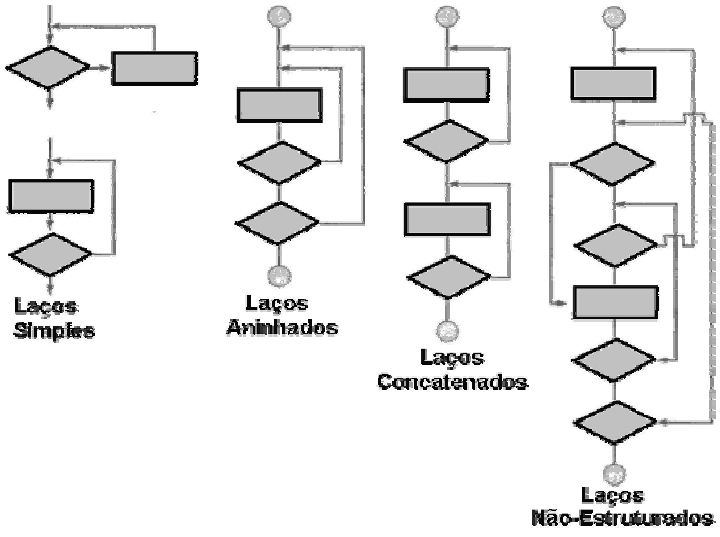
PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Laços

- ✓ Quatro categorias de laços:
  - simples (um único laço);
  - concatenados (laço após o outro);
  - aninhados (laço dentro do outro); e
  - não estruturados (desvio incondicional para outro laço).
- ✓ Teste de laços revela erros de:
  - Inicialização, Indexação, Incremento, Decremento e limites do laço.

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Laços



PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Laços

- ✓ **Laços Simples**
  - Prepare casos de teste para:
  1. Não executar o laço
  2. Uma execução do laço
  3. Duas execuções do laço
  4. m execuções do laço, tal que m < n (limite)
  5. n-1, n e n+1 execuções do laço

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Laços

- ✓ **Laços Aninhados**
  - Problema: Explosão Combinatória
  - Solução: Minimizar os casos de Testes
  1. Iniciar pelo laço mais interno; todos os demais c/ valores mínimos
  2. Abordagem laço simples para laço interno
  3. Trabalhar de “dentro para fora”
  4. Prosseguir em direção ao laço mais externo, até que todos os laços estejam testados

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Laços

- ✓ **Laços Concatenados**
  - Se variável de controle do laço 1 não afeta o laço 2, usar a abordagem de laço simples
  - caso contrário, os laços não são independentes, então usar abordagem dos laços encadeados

- ✓ **Laços Não Estruturados**
  - Devem ser reprojetados
  - Caso contrário => “Força Bruta”

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Condições

Um tipo comum de erro é cometido nos testes de condição (condições lógicas). A incidência de erros aumenta quando existem clausulas compostas, pela necessidade da utilização de operadores booleanos.

Se uma expressão condicional é composta, deve-se verificar os ramos verdadeiros e falsos de cada condição.

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Condições

Tipos de erros em condições lógicas ou expressões relacionais :

- ✓ Erro de operador booleano (incorreto, faltando, extra)
- ✓ Erro de variável
- ✓ Erro de parênteses
- ✓ Erro de operador relacional
- ✓ Erro de expressão aritmética

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Condições

Teste de Domínio de Condições:

Para uma expressão do tipo

E1 <Operador Relacional> E2

três testes são necessários (supondo-se que o operador relacional está incorreto e que E1 e E2 estão corretos):

- ✓ E1 > E2
- ✓ E1 < E2
- ✓ E1 = E2.

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Estrutural
Teste de Condições

Teste de Domínio de Condições:

E1 <Operador Relacional> E2

Para detectar erros em E1 e E2, deve-se testar a menor diferença possível em entre E1 e E2.

Para uma expressão booleana com “n” variáveis, todos os 2^n testes possíveis serão necessários, o que torna o teste viável apenas para expressões com poucas variáveis.

PUC CAMPINAS
Engenharia de Software

TESTE FUNCIONAL

(ou CAIXA PRETA)

PUC CAMPINAS
Engenharia de Software

Teste Funcional (caixa preta)

Abordagem macroscópica, baseia-se na especificação para derivar os casos de teste

✓ Foco nos Requisitos Funcionais

Objetivos:

- Verificar se uma entrada válida produz uma saída correta (esperada);
- Validar os requisitos funcionais, através da interface do software;
- Verificar a operacionalidade das funções.

PUC CAMPINAS
Engenharia de Software

Teste Funcional (caixa preta)

✓ Útil para as categorias (classes) de defeitos:

- Funções incorretas ou ausentes
- Defeitos nas Interfaces (internas e externas)
- Erros de dados ou bases de dados externas
- Inicialização e terminação
- Sensibilidade do sistema a certos valores de entrada
- Volume e taxa de dados que o sistema suporta

PUC CAMPINAS
Engenharia de Software

Teste Funcional (caixa preta)

✓ Técnicas Utilizadas:

- Particionamento de Equivalência
- Análise de Valores Limites
- Grafos de Causa-Efeito
- “Error Guessing”
- Teste de Cenários (Roteiros)

PUC CAMPINAS
Engenharia de Software

Teste Funcional (caixa preta)

Questões que devem ser respondidas:

- Funções estão Válidas?
- Quais classes de entradas caracterizarão bons casos de teste?
- Sistema é sensível a certos valores de entrada?
- Qual o comportamento do sistema nos limites e fronteiras das classes de entrada?
- Quais volumes e taxas de dados o sistema suporta?
- Qual efeito produzido pelo software com algumas combinações específicas de dados

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

- ✓ Divide o domínio de entrada do programa em **classes de dados** (Classes de Equivalência)
- ✓ Os casos de teste serão derivados a partir das classes de equivalência definidas
  - O valor representativo de uma classe de equivalência é equivalente a outro da mesma classe
- ✓ Dividir o Domínio de Saída em classes de equivalência e gerar uma tabela relacionando classes de entrada com classes de saída

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

- ✓ Etapas da Técnica:
  1. Identificar as classes de equivalência de Entrada
  2. Refinar as classes de Entrada
  3. Identificar as classes de Saída
  4. Refinar as Classes de Saída
  5. Relacionar as Classes de Entrada e de Saída
  6. Definir os casos de teste

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Etapa1: Identificar as Classes de Equivalência

- ✓ Identificar, através da especificação dos dados de entrada, as classes de equivalência
- ✓ Caracterizar pelo menos 2 grupos de valores de entrada para cada classe:
  - **Válidas**: Valores Esperados pelo programa
  - **Inválidas**: Entradas Inválidas ou inesperadas

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Etapa 1: (Continuação)

- ✓ Classes Geradas versus Tipo da Entrada

Tipo da Entrada	*Válida*	*Inválida 1*	*Inválida 2*
Intervalo	X	X	Talvez
Valor	X	X	X
Lógica	X	X	-
Conjunto	x	X	-

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Etapa 2: Refinamento das Classes de Entrada

- ✓ Se há razão para acreditar que elementos de uma mesma classe são tratados de forma diferente pelo programa, divida esta classe em menores
- ✓ Particionar Classes cujos atributos são considerados elementos chave para o item em teste

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Etapa 3: Identificar Classes de Saída

- ✓ Mapear todas as possíveis classes de saídas geradas pelo item em teste

Etapa 4: Refinar as Classes de Saída

- ✓ Verificar se alguma classe de saída merece ser subdividida em classes menores
  - Ex: Código Genérico de Erro pode ser desmembrado em outros códigos

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Etapa 5: Relacionar Classes de Entrada e Classes de Saída

- ✓ Criar um mapeamento entre as classes de entrada e as classe de saída esperadas

Etapa 6: Derivar os Casos de Teste

- ✓ Atribuir um valor de entrada para cada classe de entrada (válidas e inválidas)
- ✓ Elaborar casos de teste para as combinações das classes válidas

PUC CAMPINAS
Engenharia de Software

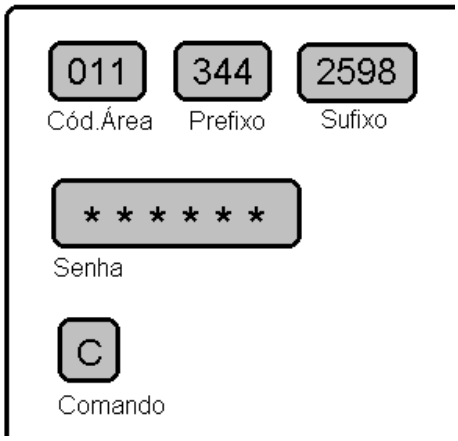
Técnicas de Teste Funcional Particionamento de Equivalência

Etapa 6: Derivar Casos de Teste

- ✓ Elaborar casos de teste para todas as classes inválidas
- ✓ Derivar casos de teste que cubram, de uma só vez, o maior número de classes válidas/inválidas (**Minimização e Inclusão**)

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência



PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Exemplo: **Condições**

Código de área	vazio ou três dígitos numéricos que iniciem por “zero”
Prefixo	três dígitos numéricos que não iniciem por “zero”
Sufixo	quatro dígitos numéricos
Senha	seis dígitos alfanuméricos
Operação	c=cheque, d=depósito, e=extrato

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Exemplo: **Caso de teste**

Código de área

- ✓ Condição booleana (estar ou não presente)
- ✓ Conjunto de valores de 000 a 099

Prefixo

- ✓ Conjunto de valores de 100 a 999

Sufixo

- ✓ Conjunto de valores de 0000 a 9999

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional Particionamento de Equivalência

Exemplo: **Caso de teste**

Senha

- ✓ Condição booleana (estar ou não presente)
- ✓ Conjunto de 6 caracteres alfanuméricos (caracteres especiais presentes ?)
- ✓ Conjunto de 1 a 5 caracteres
- ✓ Conjunto vazio

Comando

- ✓ Entrada de 1 caractere válido (c,d,e)
- ✓ Entrada de 1 caractere inválido (demais, inclusive ESC e F1)

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional
Análise dos Valores de Fronteira

- ✓ Complementa a técnica de Particionamento.
- ✓ Grande número de erros tendem a ocorrer nas fronteiras da classes de equivalência.
- ✓ Testar os limites, limites + a (próximo ao limite interno) e limites - a (próximo ao limite externo) de cada classe de equivalência não pontual (cujo domínio é dado por mais de um valor).

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional
Análise dos Valores de Fronteira

Classes de Equivalência que devem ter valores de fronteira analisados testados

- Conjunto de Valores: Testar todos que possível
- Faixa de Valores: limite, limite + a e limite - a
- Estruturas de Dados (Ex: vetor com 100 posições) Testar os valores armazenados nos índices que representam os limites da estrutura

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional
Técnica de “Error Guessing”

- ✓ Pessoas “farejadoras” (“smelling errors”)
  - Intuição e Experiência
- ✓ Enumerar listas de possíveis erros e se situações tipicamente propensas a erros
- ✓ Descrever casos de teste baseados nessas listas

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional
Técnica de “Error Guessing”

Exemplo: Teste de uma Rotina de Ordenação

CT 1. Entrada Vazia
CT 2. Entrada c/ 1 elemento
CT 3. Entrada c/ todos os elementos iguais
CT 4. Entrada c/ n+1 elementos
CT 5. Entrada já ordenada

PUC CAMPINAS
Engenharia de Software

Técnicas de Teste Funcional
Técnica de Cenários (Roteiros)

- ✓ Complementa as técnicas anteriores.
- ✓ Reaproveitar os casos de teste já gerados.
- ✓ Agrupar os Casos de Teste em roteiros que reproduzam um conjunto de transações do mundo real.
- ✓ Revela-se muito útil em teste de Integração, Aceitação e de Sistema
- ✓ Ex: Sistema Bancário
  - CT #1: Abrir a Conta - CT #2: Depósito
  - CT #3: Retirada- CT #4: Fechar a Conta

PUC CAMPINAS
Engenharia de Software

Teste Funcional x Teste Estrutural

Uma Possível Estratégia de Teste (Meyers)

1. Há combinações de entradas ou especificação pobre =>grafo causa-efeito
2. Aplicar Particionamento de Equivalência
3. Sempre Use Análise de Valores de Fronteira
4. Utilizar “Error Guessing”
5. Verificar Cobertura obtida pelo passos 1-4 dado um critério estrutural. Se necessário gerar mais alguns casos de teste

PUC CAMPINAS
Engenharia de Software

Projeto de Casos de Teste

- ✓ Teste Funcional ou Estrutural:

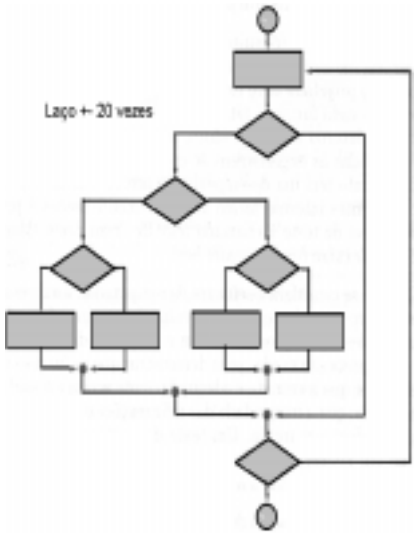
  Eis a questão?

- ✓ Paradoxo do Pesticida:

  São Técnicas complementares pois revelam erros de categorias diferentes

PUC CAMPINAS
Engenharia de Software

Projeto de Casos de Teste
Por que não usar só Caixa-Branca?



- ✓ Um teste caixa-branca que cobrisse todo o software garantiria a isenção de erros?
- ✓ Não há garantia, além do que o custo é inviável. Por exemplo, um programa Pascal com 100 linhas de código, um único loop que pode ser executado até 20 vezes: 10^{14} caminhos a serem testados.
- ✓ Se executarmos um caminho a cada 5 minutos (sem parar), levará 1 bilhão de anos.

PUC CAMPINAS
Engenharia de Software

Projeto de Casos de Teste
Por que não usar só Caixa-Preta?

- ✓ Os testes de caminho descobrem erros de projeto que não estão claros na Definição de Requisitos.
- ✓ Erros de digitação são aleatórios.

"Os bugs se escondem pelos cantos e congregam-se nas fronteiras" Beizer.

PUC CAMPINAS
Engenharia de Software

Projeto de Casos de Teste

Teste Funcional

1.Abordagem Macroscópica

2.Testes na Entrada
- Operacionalidade Funções
- Entrada Apropriadas
- Saídas Corretas
- Integridade Externa

3. Teste Exaustivo

Todas Entradas Possíveis

Ex: Entrada de 10 caracteres
- 2^{80} entradas diferentes
- 1 CT/ms => dobro da idade estimada do universo

Teste Estrutural

1. Abordagem Detalhada

2. Teste dos Detalhes Procedimentais
- Teste de Caminhos Lógicos

3. Teste Exaustivo

Todos os Caminhos Possíveis

Ex: Programa 1 laço (20 vezes), contendo dois if-then-else encadeados
- 10^{14} caminhos
- 1 CT/5 min => 1 bilhão anos

Verificação

- ✓ Processo para assegurar correção, completitude e consistência do produto implementado, em cada fase e entre fases consecutivas do ciclo de desenvolvimento de software.

Fonte: IEEE Std 729, *Standard Glossary of Software Engineering Terminology*

Validação

- ✓ Atividades que, ao final do ciclo de vida de desenvolvimento de software, testa se o software foi construído em conformidade com os requisitos desejados pelo usuário.
- ✓ Também deve ser aplicada nas fases preliminares

Fonte: IEEE Std 729, *Standard Glossary of Software Engineering Terminology*

PUC CAMPINAS
Engenharia de Software

Atividades de VV&T

- ✓ **Verificação**: Estamos construindo o software de maneira correta?
- ✓ **Validação**: Estamos construindo o software correto (aquele que é desejado pelo usuário)?
- ✓ **Teste:** Examinar o comportamento do produto com um conjunto de amostras pertencentes ao domínio de entrada do programa

PUC CAMPINAS
Engenharia de Software

Atividades de VV&T

PUC CAMPINAS
Engenharia de Software

Planejamento de Teste

- ✓ Por que Planejar o Teste?
  - Teste pode consumir mais 50% do esforço em alguns projetos de teste
  - Aumentar a Qualidade do Teste e do Software
  - Controlar os Gastos
- ✓ Planejamento de Teste deve preocupar-se:
  - Inicialmente com a definição de Padrões e Normas
  - Secundariamente, definição do material que será gerado pelo teste (testing work products)

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Validação (ou Aceitação)

- ✓ Encerrado o teste de integração, o software é um "pacote" integrado.
- ✓ Teste de aceitação é bem sucedido quando o software funciona de acordo com as expectativas (razoáveis) do usuário
- ✓ Que são as expectativas razoáveis do usuário???
  - Definidas nos Contratos de Requisitos
  - Devem ser devidamente gerenciadas através de técnicas de gerenciamento de requisitos

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Validação (ou Aceitação)

Foco: Série de **testes funcionais** que demonstram a conformidade com os requisitos do software

- ✓ Assegurar o atendimento dos Requisitos Funcionais (logo => **validação**)
- ✓ Assegurar Documentação Correta e Inteligível
- ✓ Outras Características Importantes:
  - Transportabilidade, Compatibilidade
  - Recuperação de Erros, Manutenibilidade, etc...

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Validação (ou Aceitação)

Possíveis Resultados do Teste de Aceitação:

- ✓ Aceitação pelo usuário
- ✓ Desvios ou Não-Conformidade
  - Gera uma Lista de deficiências
  - Estágio do projeto geralmente impede correções dentro do prazo
  - Negociar novos Prazos e, se for o caso, recursos ($$)

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Validação (ou Aceitação)

Técnicas Utilizadas

Teste Alfa:

- ✓ Realizado no ambiente de desenvolvimento (“look over the shoulders”)
- ✓ Desenvolvedor registra erros e problemas
- ✓ Ambiente controlado pelo testador

Teste Beta:

- ✓ Conduzido no ambiente do cliente/usuário
- ✓ Cliente relata os problemas (verdadeiros e frutos da imaginação) ao desenvolvedor

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Validação (ou Aceitação)

Outras Técnicas Utilizadas

Aderência a padrões

- ✓ Checklist são definidos para validar:
  - padrões de Interface
  - Telas
  - Relatórios
- ✓ Revisão gramatical e de terminologia utilizada

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Validação (ou Aceitação)

Técnica de Revisão de Configuração:

- ✓ Assegurar que todos os elementos do software foram desenvolvidos, catalogados e existe documentação detalhada para dar suporte à manutenção
- ✓ Também chamada de “Auditoria Geral”
- ✓ Prática Comum nas Equipes de homologação

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Sistema

Foco: **Verificar** a relação entre vários elementos do sistema (hardware, software, banco de dados, etc)

- ✓ Software é apenas um componente de grandes sistemas computacionais
- ✓ Teste fora do escopo exclusivo da Engenharia de Software
- ✓ Pode antecipar problemas que só aparecerão quando o software estiver implantado em ambiente de produção

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Sistema

Abordagem Prática:

- ✓ Levantar classes de problemas que podem ser gerados em função de outros elementos
- ✓ Simulação

1. Projetar formas de tratamento de erros
2. Projetar teste para simular as classe de problemas
3. Registrar os resultados (**evidências** de sucesso)

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Sistema

Recovery Testing (Teste de Recuperação)

Foco: Forçar o sistema a falhar, em relação à diversos aspectos do sistema computacional e verificar a resposta (em termos de recuperação do sistema)

- ✓ Recuperação, Reinicialização, Agentes de Notificação
- ✓ Sistema Tolerante a Falhas

Security Testing (Teste de Segurança)

Foco: verificar todos aspectos ligados à segurança do sistema.

- ✓ Níveis de usuário
- ✓ Invasão Externa
- ✓ Extremamente importante em ”Internet Systems”
  - Educação à Distância, Comércio Eletrônico
- ✓ Testador deve agir como “hacker” (burlar a segurança)

PUC CAMPINAS
Engenharia de Software
Estratégias de Teste
Teste de Sistema

Security Testing (Teste de Segurança)

Desde que haja tempo e recursos, um bom teste de segurança acabará por penetrar o sistema. A tarefa do projetista é fazer com que o acesso custe mais do que o valor da informação que será obtida.

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Sistema

Stress Testing (Teste de Fadiga)

Foco: Verificar as respostas do sistema quando submetido a anormalidades (em quantidade, freqüência e volume)

- ✓ Aumentar o número a taxa de interrupções
- ✓ Entradas em quantidade fora do normal
- ✓ Requerer uso máximo de memória (real e virtual)
- ✓ Manipular arquivos grandes e discos cheios

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Teste de Sistema

Performance Testing (Teste de Desempenho)

Foco: Verificar requisitos de performance em sistema de tempo real

- ✓ Pode ser conduzida conjuntamente com o teste de stress
- ✓ Requer instrumentação e manipulação de hardware e software
- ✓ Importante em software de arquitetura multi-camadas (thread testing), software de tempo real e software embarcado.

PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Quando concluir os testes ?

O teste pode mostrar a presença, mas nunca a ausência de erros no software. E. Dijkstra, 1969.

Como obter critérios para a conclusão de testes?

Resposta: Você jamais terá completado a atividade de teste; a carga simplesmente se transfere do desenvolvedor para o usuário, pois toda vez que o software é executado, ele está sendo testado em relação a um novo conjunto de dados.

Resposta: A atividade de teste só se esgotaria quando o projeto estiver sem tempo (prazo) e dinheiro (orçamento).

PUC CAMPINAS
Engenharia de Software

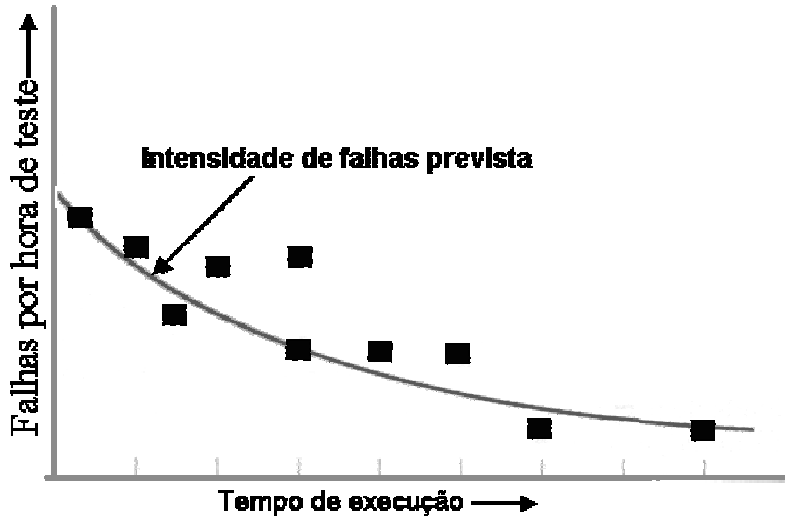
Estratégias de Teste
Quando concluir os testes ?

Critérios mais rigorosos para que, no mínimo, possamos definir um mínimo aceitável de teste.

- ✓ Modelos de confiabilidade (por exemplo o modelo logarítmico do tempo de execução de Poisson)
- ✓ Medidas e métricas
  - Número cumulativo de falhas encontradas por unidade de tempo de teste.
  - Número cumulativo de falhas encontradas durante todo o tempo de teste.

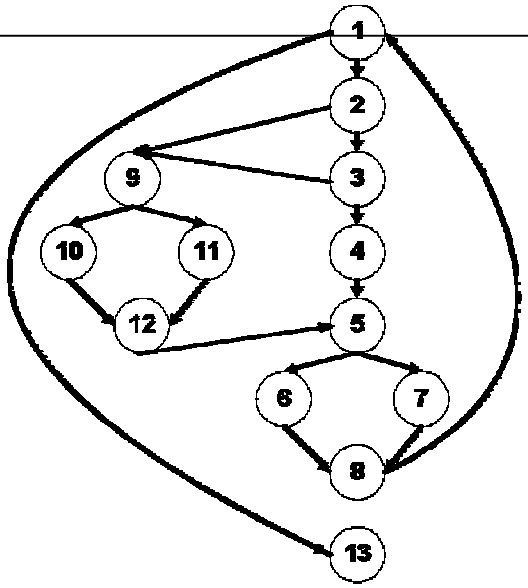
PUC CAMPINAS
Engenharia de Software

Estratégias de Teste
Quando concluir os testes ?



PUC CAMPINAS
Engenharia de Software

Exercício



PUC CAMPINAS

Engenharia de Software

Exercício

1. Desenhar o Grafo de Fluxo de Controle; indicar nele as seguintes entidades:
   - Arcos
   - Nós
   - Nós Predicativos
   - Regiões
2. Calcular o número máximo de caminhos independentes V(G), usando as três fórmulas conhecidas.
3. Descrever a seqüência dos caminhos independentes existentes, usando a numeração dos nós existentes em cada caminho.

Exercício para ser resolvido em equipe de 4 alunos. Entregar no final da aula.