



Técnica de Teste Funcional

PPGCC12-Teste de Software

Reginaldo Ré

Aula 04



Recapitulando



Aula Anterior

- Diferentes **técnicas** e **critérios** de teste existem para auxiliar na atividade de teste.
 - Basicamente, os testes podem ser classificados em **teste caixa-preta** (teste funcional) ou **teste caixa-branca** (teste estrutural).
 - Contemplam diferentes perspectivas do software: **aspecto complementar!!!!**

Técnica Funcional



Técnica Funcional

- Também conhecida como **Técnica Caixa-Preta**
 - Considera o produto em teste como uma caixa da qual só se conhece a entrada e a saída (sem conhecimento da parte interna).
- Baseia-se na **especificação do software** para derivar os requisitos de teste.
 - Aborda o software de um ponto de vista macroscópico.
 - Não se preocupa com detalhes de implementação.

Passos da Técnica Funcional

- Passos básicos para aplicar um critério de teste funcional:
 - A especificação de requisitos é analisada.
 - Entradas **válidas** são escolhidas para determinar se o produto em teste comporta-se corretamente.
 - Entradas **inválidas** são escolhidas para verificar se estas são detectadas e manipuladas adequadamente.
 - Os casos de testes são construídos (saídas são determinadas para cada entrada).
 - O conjunto de teste é executado e as saídas obtidas são comparadas com as saídas esperadas.
 - Um relatório é gerado para avaliar o resultado dos testes.

Aplicabilidade da Técnica Funcional

- Por ser independente da implementação, critérios da técnica funcional podem ser utilizados em todas as **fases de teste**.
- A complexidade de aplicação aumenta em cada fase.
 - Unidade
 - Integração
 - Sistema

Critérios de Teste Funcionais (1/2)

- Critérios de teste funcionais mais conhecidos:
 - **Particionamento em Classes de Equivalência**
 - Divide o domínio de entrada (e de saída) de um programa em **classes de equivalência**, a partir das quais derivam-se os casos de teste.
 - **Análise do Valor Limite**
 - Complementa o critério Particionamento de Equivalência, exigindo casos de teste nos **limites** (fronteiras) de cada classe de equivalência.
 - **Grafo de Causa-Efeito**
 - Verifica o **efeito combinado** de dados de entrada.
 - Causas (condições de entrada) e efeitos (ações) são identificados e combinados em um grafo.
 - Tabela de Decisão -> Casos de Teste
 - *Error guessing*
 - ≡ ◦ Projeto baseado na intuição e experiência

Critérios de Teste Funcionais (2/2)

- Existem outros:
 - Teste funcional sistemático
 - *Syntax testing*
 - *State transtion testing*
 - *Graph matrix*
 - Tabelas de decisão

Particionamento em Classes de Equivalência (1/7)

- Critério utilizado para reduzir o número de casos de teste, procurando garantir uma **boa cobertura** do código do produto em teste.
- Empregado **intuitivamente** pelos programadores mesmo sem conhecer o critério.

Particionamento em Classes de Equivalência (2/7)

- Exemplo: Parte de um Sistema de Recursos Humanos que determina contratações com base na idade dos candidatos.

| | |
|---------|-------------------------------------|
| 0 - 16 | Não empregar |
| 16 - 18 | Pode ser colaborador tempo parcial |
| 18 - 55 | Pode ser colaborador tempo integral |
| 55 - 99 | Não empregar |

- Como derivar **casos de teste**?

Particionamento em Classes de Equivalência (3/7)

Considere a seguinte implementação

```
1  if (idade == 0) empregar = "NAO";  
2  if (idade == 1) empregar = "NAO";  
3  ...  
4  if (idade == 15) empregar = "NAO";  
5  if (idade == 16) empregar = "PAR";  
6  if (idade == 17) empregar = "PAR";  
7  if (idade == 18) empregar = "INT";  
8  if (idade == 19) empregar = "INT";  
9  ...  
10 if (idade == 53) empregar = "INT";  
11 if (idade == 54) empregar = "INT";  
12 if (idade == 55) empregar = "NAO";  
13 if (idade == 56) empregar = "NAO";  
14 ...  
15 if (idade == 98) empregar = "NAO";  
16 if (idade == 99) empregar = "NAO";
```

- Neste caso, a única forma de testá-lo adequadamente seria executar o módulo com valores de idade de **0..99**.
- Caso haja tempo suficiente, esse é o melhor teste a ser realizado!
- O problema é que da forma como o código anterior foi implementado, a execução de um dado caso de teste não diz nada a respeito da execução do próximo.

Particionamento em Classes de Equivalência (4/7)

Considere a seguinte (melhor) implementação

```
1  if (idade >= 0 && idade <= 16)
2      empregar = "NAO";
3  if (idade >= 16 && idade <= 18)
4      empregar = "PAR";
5  if (idade >= 18 && idade <= 55)
6      empregar = "INT";
7  if (idade >= 55 && idade <= 99)
8      empregar = "NAO";
```

- Dada essa implementação, fica claro que não é necessário testar para todos os valores 0, 1, 2, \dots , 14, 15 e 16, por exemplo.
- Apenas um **conjunto de valores** precisa ser testado.
 - **Quais seriam esses valores?**

Particionamento em Classes de Equivalência (5/7)

- Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado.
- O mesmo se aplica para os demais intervalos de dados.
- Tais intervalos determinam o que é chamado de **Classes de Equivalência**.
- Qualquer valor no intervalo de uma classe é considerado **equivalente** em termos de teste.
 - Se um caso de teste de uma classe de equivalência revela um erro, qualquer caso de teste da mesma classe também revelaria e vice-versa.

Particionamento em Classes de Equivalência (6/7)

- Esse critério assume que existe uma indicação precisa das classes de equivalência.
- É assumido que não existe algo estranho como:

```
1  if (idade >= 0 && idade <= 16)
2      empregar = "NAO";
3  if (idade >= 16 && idade <= 18)
4      empregar = "PAR";
5  if (idade >= 18 && idade <= 41)
6      empregar = "INT";
7  // início comando estranho
8  if (idade == 42 && nome == "Fulano")
9      empregar = "INT-DIF";
10 if (idade == 42 && nome != "Fulano")
11     empregar = "INT";
12 // fim comando estranho
13 if (idade >= 55 && idade <= 99)
14     empregar = "NAO";
```


Particionamento em Classes de Equivalência (7/7)

- Observe que esse critério de teste reduz o número de casos de teste de 100 para 4 (um para cada classe de equivalência).
- **Devem ser considerados casos de teste inválidos!**

Passos para Aplicação do Particionamento (1/10)

- Identificar as classes de equivalência (requisitos de teste do critério).
 - Condições de entrada.
 - Classes válidas e inválidas.
- Definir os casos de teste.
 - Enumerar as classes de equivalência.
 - Criar casos de teste para as classes de equivalência válidas.
 - Criar um caso de teste para cada classe de equivalência inválida.
 - Entradas inválidas são grandes fontes de defeitos!
- Casos de teste adicionais podem ser criados caso haja tempo e dinheiro suficientes.

Passos para Aplicação do Particionamento (2/10)

Exemplo: Programa String

O programa string solicita do usuário um inteiro positivo no intervalo entre 1 e 20 e então solicita uma cadeia de caracteres desse comprimento.

*Após isso, o programa solicita um caracter e retorna a posição na cadeia em que o caracter é encontrado pela primeira vez ou uma mensagem indicando que o caracter não está presente na cadeia. **O usuário tem a opção de procurar por vários caracteres - um de cada vez!!***

Passos para Aplicação do Particionamento (3/10)

Exemplo: Programa String

- Identificar as condições de entrada

| Condição de entrada | Classes de equivalência válidas | Classes de equivalência inválidas |
|---------------------------------|---------------------------------|-----------------------------------|
| Tamanho da cadeia (T) | | |
| A cadeia de caracteres (CC) | | |
| O caractere a ser procurado (C) | | |
| Procurar outros caracteres? (O) | | |

Passos para Aplicação do Particionamento (4/10)

Exemplo: Programa String

- Indentificar as classes válidas e inválidas

| Condição de entrada | Classes de equivalência válidas | Classes de equivalência inválidas |
|---------------------------------|---------------------------------|-----------------------------------|
| Tamanho da cadeia (T) | $1 \leq T \leq 20$ | $T < 1$ e $T > 20$ |
| A cadeia de caracteres (CC) | $CC = T$ (tamanho) | $CC \neq T$ (tamanho) |
| O caractere a ser procurado (C) | Pertence Não Pertence | |
| Procurar outros caracteres? (O) | S N | Qualquer outro caractere |

Passos para Aplicação do Particionamento (5/10)

Exemplo: Programa String

- Criar casos de teste

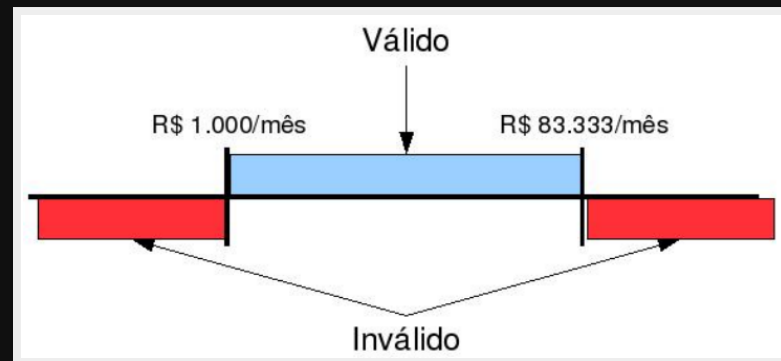
| Condição de entrada | Classes de equivalência válidas | Classes de equivalência inválidas |
|---------------------------------|---------------------------------|-----------------------------------|
| Tamanho da cadeia (T) | $1 \leq T \leq 20$ (1) | $T < 1$ (2) e $T > 20$ (3) |
| A cadeia de caracteres (CC) | $CC = T$ (4) | $CC \neq T$ (5) |
| O caractere a ser procurado (C) | Pertence (6) Não Pertence (7) | |
| Procurar outros caracteres? (O) | S (8) N (9) | Qualquer outro caractere (10) |

- Conjunto de casos de teste:
- $T = \{(<6, \text{"alface"}, a, N>, 1), (<6, \text{"alface"}, x, N>, \text{não pertence}), (<6, \text{"alface"}, a, S, c, N>, 1, 5), (-2, T \text{ inválido}), (<6, \text{"alfa"}, >, \text{string inválida}), (<6, \text{"alface"}, a, X>, 1, \text{entrada inválida}), (25, T \text{ inválido})\}$

Passos para Aplicação do Particionamento (6/10)

Exemplo 2: Sistema de hipotecas

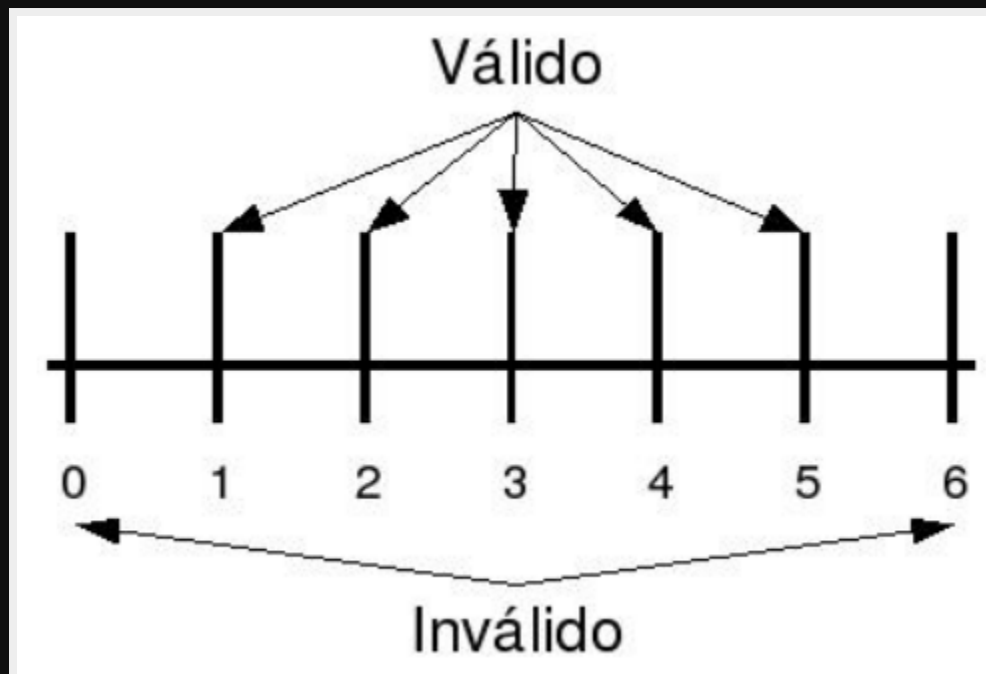
- Ilustra a definição de classes de equivalência para diferentes tipos de dados.
- **Classes para Dados Contínuos** (renda mensal para hipoteca deve estar entre de R\$1.000 a R\$83.333):
 - São definidas duas classes inválidas e uma válida.
 - Para a classe válida poderia ser escolhido R\$1.342/mês.
 - Para as classes inválidas poderiam ser escolhidos R\$123/mês e R\$90.000/mês



Passos para Aplicação do Particionamento (7/10)

Exemplo 2: Sistema de hipotecas

- **Classes para Dados Discretos** (hipotecas de 1 a 5 casas):
 - São definidas duas classes inválidas e uma válida.
 - Para a classe válida poderia ser escolhido 2.
 - Para as classes inválidas poderiam ser escolhidos -2 e 8.



Passos para Aplicação do Particionamento (8/10)

Exemplo 2: Sistema de hipotecas

- Classes para Dados Simples (somente hipoteca para pessoa física é permitido):
 - São definidas uma classe inválida e uma válida.
 - Para a classe válida poderia ser escolhida uma pessoa qualquer.
 - Para a classe inválida deve ser escolhida uma companhia ou associação.

Passos para Aplicação do Particionamento (9/10)

Exemplo 2: Sistema de hipotecas

- **Classes para Dados de Múltipla Escolha** (três tipos de hipoteca são válidas: condomínio, sobrado e casa térrea):
 - Para o intervalo válido pode-se escolher: condomínio, sobrado ou casa térrea.
 - Escolher somente um ou os três? Depende da criticalidade do programa em teste. Se forem poucos itens vale a pena selecionar um de cada.
 - O mesmo para a classe inválida.

Passos para Aplicação do Particionamento (10/10)

Exemplo 2: Sistema de hipotecas

- Devido ao número de condições de entrada, não há tempo para a criação de um caso de teste para cada classe válida.
 - Solução: Criar o menor número possível de casos de teste que cubra todas as classes válidas.
 - Criar um caso de teste para cada classe inválida.

| Renda | # Casas | Aplicante | Tipo | Resultado |
|--------|---------|-----------------|-------------|-----------|
| 5.000 | 2 | Pessoa Física | Condomínio | Válido |
| 100 | 1 | Pessoa Física | Casa térrea | Inválido |
| 90.000 | 1 | Pessoa Física | Casa térrea | Inválido |
| 1.342 | 0 | Pessoa Física | Condomínio | Inválido |
| 1.342 | 6 | Pessoa Física | Condomínio | Inválido |
| 1.342 | 1 | Pessoa Jurídica | Sobrado | Inválido |
| 1.342 | 1 | Pessoa Física | Suplex | Inválido |

Aplicabilidade e Limitações do Particionamento

- Reduz significativamente o número de casos de teste em relação ao teste exaustivo.
- Mais adequado para o teste de produtos com domínios de entrada divididos em intervalos ou conjuntos.
- Assume que os valores dentro da mesma classe são equivalentes (isso nem sempre é verdade!).
 - Importante empregar outros critérios de teste!!
- Aplicável em todas as fases de teste: unidade, integração e sistema.

Análise do Valor Limite (1/3)

- Complementa o critério Particionamento de Equivalência, exigindo casos de teste nos **limites** (fronteiras) de cada classe de equivalência.
- Considerando o exemplo utilizado anteriormente:

| | |
|---------|-------------------------------------|
| 0 - 16 | Não empregar |
| 16 - 18 | Pode ser colaborador tempo parcial |
| 18 - 55 | Pode ser colaborador tempo integral |
| 55 - 99 | Não empregar |

- Observe que os limites aparecem em duas classes de equivalência (16 por exemplo).

Análise do Valor Limite (2/3)

- As condições anteriores, na verdade, deveriam ser escritas como:

| | |
|-----------------------------|-------------------------------------|
| $0 \leq \text{idade} < 16$ | Não empregar |
| $16 \leq \text{idade} < 18$ | Pode ser colaborador tempo parcial |
| $18 \leq \text{idade} < 55$ | Pode ser colaborador tempo integral |
| $55 \leq \text{idade} < 99$ | Não empregar |

- ou

| | |
|--------------------------------|-------------------------------------|
| $0 \leq \text{idade} \leq 15$ | Não empregar |
| $16 \leq \text{idade} \leq 17$ | Pode ser colaborador tempo parcial |
| $18 \leq \text{idade} \leq 54$ | Pode ser colaborador tempo integral |
| $55 \leq \text{idade} \leq 99$ | Não empregar |

Análise do Valor Limite (3/3)

Considere a seguinte implementação

```
if (idade >= 0 && idade <= 15)
    empregar = "NAO";
if (idade >= 16 && idade <= 17)
    empregar = "PAR";
if (idade >= 18 && idade <= 54)
    empregar = "INT";
if (idade >= 55 && idade <= 99)
    empregar = "NAO";
```

- Valores limites a serem considerados: {-1, 0}, {15, 16}, {17, 18}, {54, 55} e {99, 100}

Passos para a aplicação da AVL (1/3)

- Identificar as classes de equivalência (requisitos de teste do critério).
- Identificar os limites de cada classe.
- Criar casos de teste para os limites escolhendo:
 - Um ponto abaixo do limite.
 - O limite.
 - Um ponto acima do limite.
- Observe que acima e abaixo são termos relativos e dependentes do valor dos dados.
 - Números inteiros: limite = 16; abaixo = 15; acima = 17.
 - Números reais: limite = \$5,00; abaixo = \$4,99; acima = \$5,01.
- Casos de teste adicionais podem ser criados dependendo dos recursos disponíveis.

Passos para a aplicação da AVL (2/3)

Prática: Programa String

- Casos de teste para AVL

| Condição de entrada | Classes de equivalência válidas | Classes de equivalência inválidas |
|---------------------------------|---------------------------------|-----------------------------------|
| Tamanho da cadeia (T) | $1 \leq T \leq 20$ | $T < 1$ e $T > 20$ |
| A cadeia de caracteres (CC) | $CC = T$ | $CC \neq T$ |
| O caractere a ser procurado (C) | Pertence Não Pertence | |
| Procurar outros caracteres? (O) | S N | Qualquer outro caractere |

- Conjunto de casos de teste:
- $T = \{ ??? \}$

Passos para a aplicação da AVL (3/3)

Aplicabilidade e limitações da AVL

- Reduz significativamente o número de casos de teste em relação ao teste exaustivo.
- Mais adequado para o teste de produtos com domínios de entrada divididos em intervalos ou conjuntos.
- Aplicável em todas as fases de teste: unidade, integração e sistema.

Considerações Finais

Pontos importantes

- A técnica funcional pode ser utilizada em **todas** as fases de teste.
- Independe do paradigma de programação utilizado.
- Eficaz em detectar determinados tipos de erros.
 - Por exemplo: Funcionalidade ausente.
 - Isso é: erros de omissão
- Dependente de uma **boa** especificação de requisitos.
 - Especificações descritivas e não formais.
 - Requisitos imprecisos e informais.
- Dificuldade em **quantificar** a atividade de teste.
- Não é possível garantir que partes essenciais ou críticas do software sejam executadas.
- Dificuldade de **automatização**: em geral, a aplicação é manual.

Prática: Programa Identifier

- Gerar casos de teste utilizando os critérios Particionamento em Classe de Equivalência e Análise do Valor Limite.

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.

| Identificadores válidos | Identificadores inválidos |
|-------------------------|---------------------------|
| abc12 | cont*1 |
| C4d5 | 1soma |
| dcdf | a123456 |