Critério Particionamento de Equivalência

Técnica Caixa-Preta

Auri Marcelo Rizzo Vincenzi[1], Márcio Eduardo Delamaro[2] e José Carlos Maldonado[2]

[1]Instituto de Informática
Universidade Federal de Goiás

[2]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

Organização

Introdução (1)

- Critério utilizado para reduzir o número de caso de teste procurando garantir uma boa cobertura do código do produto em teste.
- Empregado intuitivamente pelos programadores mesmo sem conhecer o critério.
- Exemplo: sistema de recursos humanos – empregar pessoas com base na idade (Copeland, 2004).

0 – 16	Não empregar.
16 – 18	Pode ser empregado tempo parcial.
18 – 55	Pode ser empregado tempo integral.
55 – 99	Não empregar.

- Como deveriam ser derivados casos de teste para o exemplo acima?

Introdução (2)

- O módulo deveria ser testado considerando as idades: 0, 1, 2, 3, 4, 5, 6, 7, 8, ..., 90, 91, 92, 93, 94, 95, 96, 97, 98, 99?

Introdução (2)

- O módulo deveria ser testado considerando as idades: 0, 1, 2, 3, 4, 5, 6, 7, 8, ..., 90, 91, 92, 93, 94, 95, 96, 97, 98, 99?
- Considere que o módulo que resolve o problema anterior tenha sido implementado como ilustrado abaixo:

```
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";
```

Introdução (3)

- Caso o programa tenha sido implementado acima, 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 acima foi implementado, a execução de um dado caso de teste não diz nada a respeito da execução do próximo.

Introdução (4)

Agora considere essa outra implementação (bem melhor) do mesmo problema:

```
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";
```

(extraído de Copeland (2004))

- Dada essa implementação, fica claro que não é necessário testar para todos os valores 0, 1, 2, $\cdots$, 14, 15 e 16, por exemplo.
- Apenas um valor precisa ser testado.
- Qual seria esse valor?

Introdução (5)

- Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado.

Introdução (5)

- 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.

Introdução (5)

- 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 **classe de equivalência**

Introdução (5)

- 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 **classe de equivalência**
- Qualquer valor no intervalo de uma classe é considerado **equivalente** em termos de teste. Assim sendo:

Introdução (5)

- 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 **classe de equivalência**
- Qualquer valor no intervalo de uma classe é considerado **equivalente** em termos de teste. Assim sendo:
  - 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.

Introdução (6)

- Tal critério de teste assume que na especificação de requisitos existe uma indicação precisa das classes de equivalência.
- Além disso, também é assumido que o programador não implementou algo estranho como ilustrado abaixo:

```
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 comado 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";
```

(extraído de Copeland (2004))

Introdução (7)

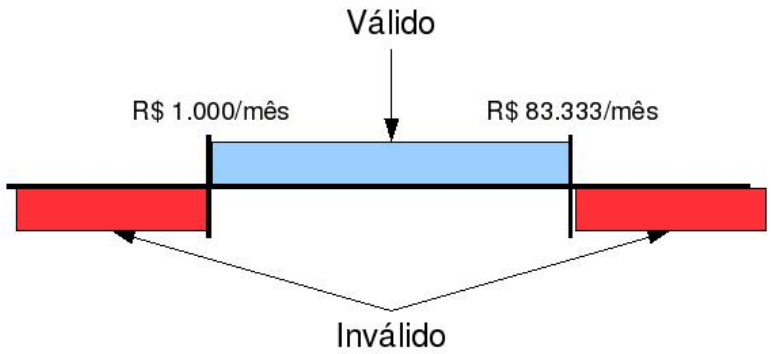
- Observe que com esse critério de teste o número de casos de teste é reduzido de 100 para 4 (um para cada classe de equivalência).
- Casos de teste inválidos devem ser considerados?
  - Projeto por contrato – *Design-by-contract* (não)
    - pré-condição
    - pós-condição
  - Projeto defensivo – `Defensive design` (sim)

Passos de Aplicação

1. Identificar as classes de equivalência (requisitos de teste do critério).
2. Criar casos de testes para as classes de equivalência válidas.
3. Criar um caso de teste para cada classe de equivalência inválida (entradas inválidas são grandes fontes de defeitos).
4. Casos de teste adicionais podem ser criados caso haja tempo e recursos suficientes.
   - Com base em sua experiência, o(a) testador(a) pode criar casos de teste adicionais.

Definição das Classes (1)

- Diferentes tipos de dados exigem diferentes tipos de classe de equivalência.
- Intervalo de dados contínuos (renda para hipoteca de R$1.000 a 83.000/mês):

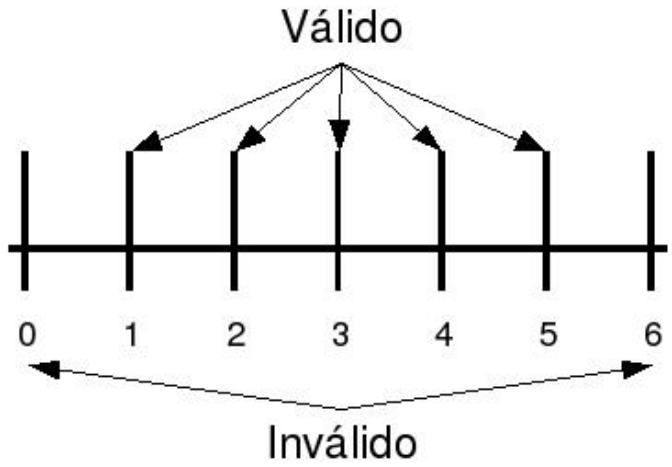


(extraído de Copeland (2004))

  - Em geral são definidas duas classes inválidas e uma válida.
  - Para a classe válida poderia ser escolhido R$1.342/mês.
  - Para as classes inválidas poderia ser: R$123/mês e R$90.000/mês.

Definição das Classes (2)

- Intervalo de dados discretos (hipotecas de 1 a 5 casas):

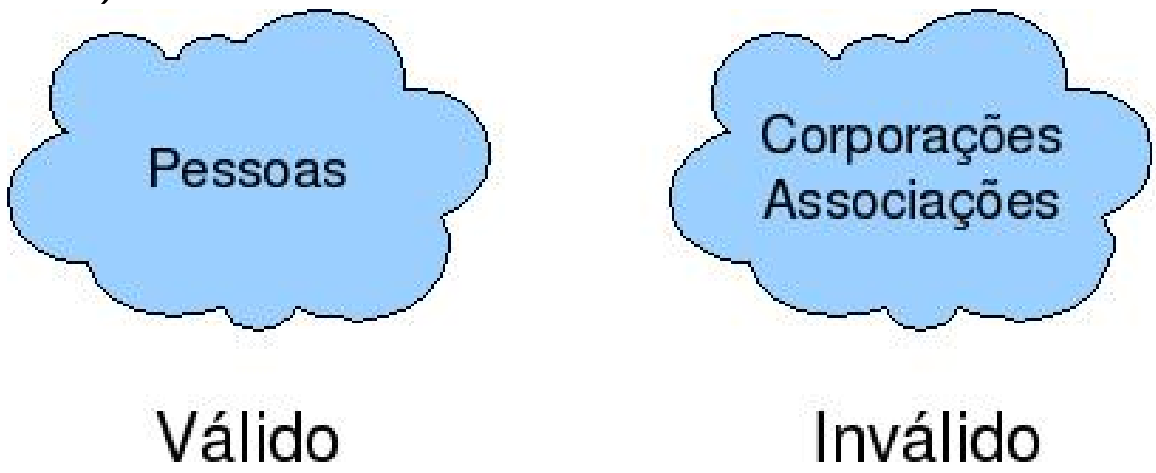


(extraído de Copeland (2004))

  - Em geral são definidas duas classes inválidas e uma válida.
  - Para a classe válida poderia ser escolhido 2.
  - Para as classes inválidas poderia ser: -2 e 8.

Definição das Classes (3)

- Intervalo de dados simples (somente hipoteca para pessoas é permitido):



(extraído de Copeland (2004))

  - Em geral 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.

Definição das Classes (4)

- Intervalo de dados de múltipla escolha (três tipos de hipoteca são válidas: condomínio, sobrado e casa térrea):

(extraído de Copeland (2004))

  - 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

Definição das Classes (5)

- Em geral, 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 cubram todas as classes válidas.
  - Criar um caso de teste para cada classe inválida.

Renda	# Moradores	Aplicante	Tipo	Resultado
$5.000	2	Pessoas	Condomínio	Válido
$100	1	Pessoas	Uma família	Inválido
$90.000	1	Pessoas	Uma família	Inválido
$1.342	**0**	Pessoas	Condomínio	Inválido
$1.342	**6**	Pessoas	Condomínio	Inválido
$1.342	1	**Corporação**	Sobrado	Inválido
$1.342	1	Pessoas	**Duplex**	Inválido

Definição das Classes (6)

- Uma abordagem adicional do critério Particionamento de Equivalência é considerar as saídas.
- O domínio de saída também é particionado em classes válidas e inválidas.
- Casos de teste que causem tais saídas são então desenvolvidos.

Programa Identifier (1)

Especificação (extraído de Maldonado et al. (2004)):

O programa deve determinar se um identificador é válido ou não em Silly Pascal (uma variante do Pascal). 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.

Exemplos de Identificadores:

abc12 (válido);
cont*1 (inválido); 1soma (inválido); a123456 (inválido)

Programa `Identifier` (2)

Classes de Equivalência:

Condições de Entrada	Classes Válidas	Classes Inválidas	
Tamanho t do identificador	$1 \leq t \leq 6$ (1)	$t < 1$ (2)	$t > 6$ (3)
Primeiro caractere c é uma letra	Sim (4)	Não (5)	
Só contém caracteres válidos	Sim (6)	Não (7)	

Exemplo de Conjunto de Teste:
$T_0 = \{$(`a1`, Valid), (`""`, Invalid), (`A1b2C3d`, Invalid), (`2B3`, Invalid), (`Z#12`, Invalid)$\}$

Outros Exemplos

- Outros exemplos do critério Particionamento de Equivalência pode ser encontrado no Capítulo 3 do livro de (Copeland, 2004).

Aplicabilidade e Limitações

- 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.
- Aplicável em todas as fases de teste: unidade, integração, sistema e aceitação.

Resumo

- Critério Particionamento de Equivalência divide o domínio de entrada em classes de equivalência.
- Possivelmente, classes válidas e inválidas devem ser consideradas (projeto defensivo).
- Simples e intuitiva para a maioria dos programadores.

Programa `cal` do Unix – Especificação

`cal [[month] year]`

"Um único parâmetro especifica o ano (`year`) a ser exibido e pode variar entre 1 e 9999; observe que o ano deve ser completamente especificado: `cal 89` não exibe o calendário do ano 1989 mas sim do ano 89.
Dois parâmetros são utilizados para denotar o mês (`month`) e o ano, sendo que o mês pode variar entre 1 e 12). Caso nenhum parâmetro seja fornecido, o mês do ano atual é exibido.
O ano se inicia em 1 de Jan.
A reforma no calendário Gregoriano (The Gregorian Reformation) ocorreu no dia 3 de setembro de 1752. Até o momento, a maioria dos países reconheceu a reforma realizada (embora poucos ainda não o tinham feito até os anos 90). Com a reforma, dez dias foram eliminados do calendário a partir da data acima exibindo um calendário diferente para o mês e ano em questão."

Com base na especificação acima, considerando o critério **Particionamento de Equivalência**, defina quais as classes de equivalência válidas e inválidas e derive casos de testes que satisfaçam o critério.

Leitura Recomendada

Mais informações sobre esse tema podem ser encontrados em:

- Seção 1, Capítulo 3 do livro de Copeland (2004).

Referências I

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

Maldonado, J. C.; Barbosa, E. F.; Vincenzi, A. M. R.; Delamaro, M. E.; Souza, S. R. S.; Jino, M. *Introdução ao teste de software*. Relatório Técnico 65 – Versão 2004-01, Instituto de Ciências Matemáticas e de Computação – ICMC-USP, disponível on-line: http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_ND_65.pdf., 2004.