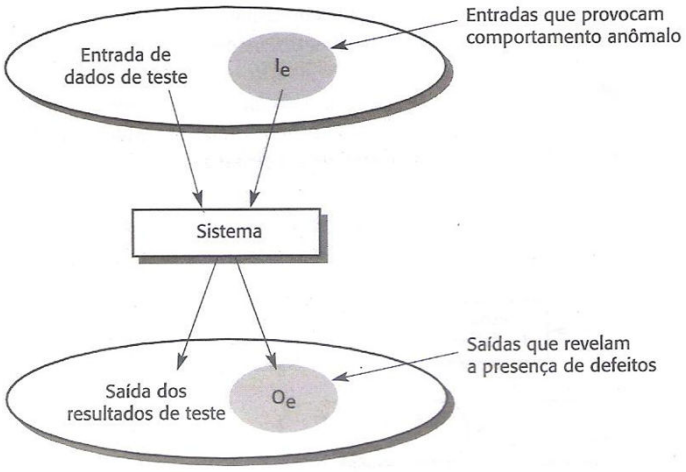
TESTE DE PARTIÇÃO DE EQUIVALÊNCIA

Prof. Rafael Dias Ribeiro

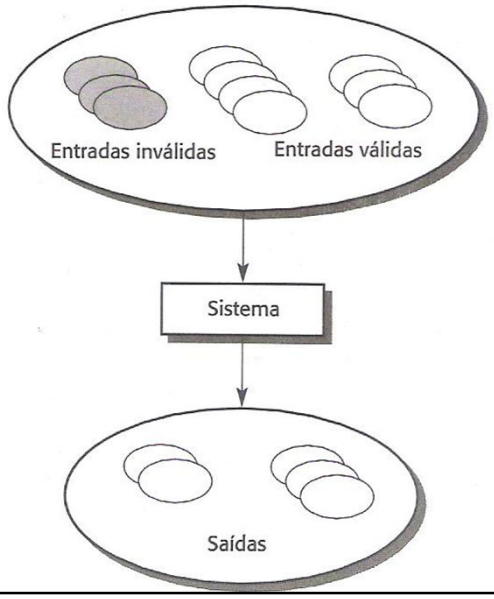
Teste de Partição de Equivalência



Teste de Partição de Equivalência

- Dados de entrada e resultados de saída caem em diferentes classes onde todos os membros de uma classe são relacionados.
- O programa se comporta de maneira equivalente para cada membro da classe.
- Casos de teste devem ser escolhidos para cada partição.

Teste de Partição de Equivalência



Teste de Partição de Equivalência

- O domínio de entrada (ou saída) do programa/função é dividido em um número finito de partições (ou classes) de equivalência
  - supõe-se que dados pertencentes a uma partição revelam as mesmas falhas
  - partições válidas e inválidas são consideradas
- Geração de testes: selecionar um ou mais dados de cada partição
- Critério de cobertura: cada partição deve ser considerada ao menos 1 vez

Teste de Partição de Equivalência – Passos:

- Decompor o programa em funções
- Identificar as variáveis que determinam o comportamento de cada função
- Particionar os valores de cada variável em classes de equivalência (válidas e inválidas)
- Especificar os casos de teste:
  - eliminar as classes impossíveis ou os casos desinteressantes
  - selecionar casos de testes cobrindo as classes válidas das diferentes variáveis
  - para cada classe inválida escolha um caso de teste que cubra 1 e somente 1 de cada vez

Definição de Classes de Equivalências

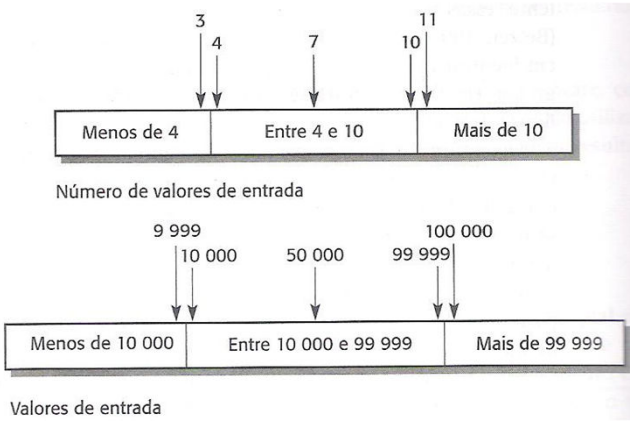
Definição da variável de entrada	Classes de equivalência
Intervalo	• Uma classe válida para valores pertencentes ao intervalo • Uma classe inválida para valores menores que o limite inferior • Uma classe inválida para valores maiores que o limite superior
Lista de valores válidos	• Uma classe válida para os valores incluídos na lista • Uma classe inválida para todos os outros valores

Definição de Classes de Equivalências

Definição da variável de entrada	Classes de equivalência
Número de valores válidos	• Uma classe válida para número de valores igual ao número previsto • Uma classe inválida para número de valores = 0 • Uma classe inválida para número de valores maior ou menor que o valor previsto
Restrições (expressão lógica; sintaxe; valor específico; compatibilidade com outras variáveis)	• Uma classe válida para os valores que satisfazem às restrições • Uma classe inválida para os outros valores

Teste de Partição de Equivalência – Exemplo:

- Uma especificação de programa aceita de 4 a 10 entradas, que são números inteiros de 5 dígitos e maiores que 10 mil.



Teste de Partição de Equivalência – Exemplo:

Ex. Especificação de uma rotina de busca.

Procedure Search (key: elem; T:elem_array; Found: in out Boolean; L in out elem_index);

Pré-condição: *T'first <= T'last*
A sequência tem pelo menos um elemento.

Pós-condição: *(Found and T(L)=key)*
O element é encontrado e referenciado por L.

(Not found and not(Exists i; T'first >= i <= T'last; T(i) = key))
O elemento não está na sequência.

Teste de Partição de Equivalência – Exemplo:

Partições de entrada:

- Entradas que estão de acordo com a pré-condição (no mínimo 1 elemento).
- Entrada onde a pré-condição não vale.
- Entradas onde o elemento chave é um elemento da sequência.
- Entradas onde o elemento chave não é um membro da sequência.

Diretrizes de teste:

- Teste com sequências que possuem somente um único valor.
- Use diferentes sequências, de diferentes tamanhos, em diferentes testes.
- Derive de maneira que o primeiro, o médio e o último elemento da sequência sejam acessados.
- Testes com sequências de comprimento zero.

Teste de Partição de Equivalência – Exemplo:

Vetor	Elemento
Valor único	Está na sequência
Valor único	Não está na sequência
Mais que 1 valor	Primeiro elemento está na sequência
Mais que 1 valor	Último elemento está na sequência
Mais que 1 valor	Elemento médio está na sequência
Mias que 1 valor	Não está na sequência

Sequência de entrada (T)	Chave (Key)	Saídas (Found, L)
17	17	Verdadeiro,1
17	0	Falso, ?
17, 29, 21, 23	17	Verdadeiro, 1
41, 18, 9, 31, 30, 16, 45	45	Verdadeiro, 7
17, 18, 21, 23, 29, 41, 38	23	Verdadeiro, 4
21, 23, 29, 33, 38	25	Falso, ?

Notas de Aula retiradas do Livro

Livro:
ENGENHARIA DE SOFTWARE
SOMMERVILLE, IAN
6 Ed.