

Estratégias de concepção de casos de teste



2015/2016
1º semestre

Testes de Caixa Branca

Por:

Cristóvão Sousa e Carla Pereira

Sumário

- ▶ Conceber casos de teste segundo uma abordagem a testes de caixa branca.

motivação dos testes de caixa branca

- ▶ Os “testers” necessitam de uma framework para:
- ▶ decidirem quais os elementos estruturais que devem seleccionar como foco para o teste
- ▶ escolher os dados de teste apropriados
- ▶ decidirem quando os esforços de teste são adequados e suficientes para poderem terminar o processo confiantes de que o software funciona adequadamente
- ▶ Mas quando parar?!

quais e quantos?

Uma das questões mais importantes na execução de um teste é caracterizar quais (e quantos) testes são suficientes para garantir qualidade do produto.

recapitulando

Input:
- Requisitos cliente
- Requisitos de Sw
- Modelos, Interface, etc..



Quantos casos de teste?

A estimativa cobre todos os requisitos?

Não me esqueci de nenhum requisito?

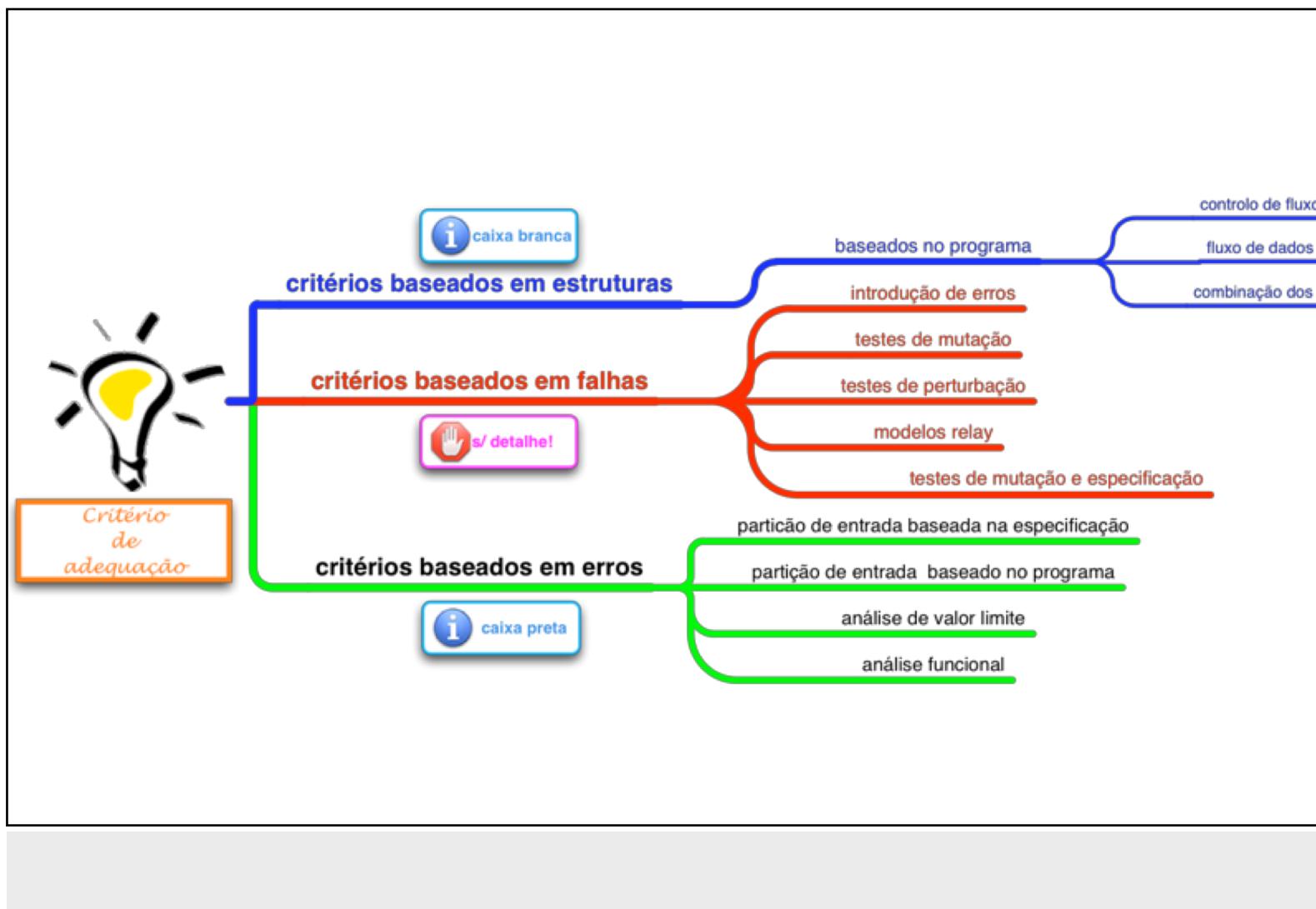
Até que ponto é a minha estimativa confiável?

critérios de adequação

- ▶ Um **critério de adequação** de dados de teste **é uma regra de paragem** – as regras deste tipo podem ser usadas para determinar se os testes realizados são suficientes ou não
- ▶ O âmbito de aplicação do critério de adequação também inclui:
- ▶ ajudar o “tester” a seleccionar as propriedades de um programa que deve focar durante o teste;
- ▶ ajudar o “tester” a seleccionar o conjunto de dados de teste para um programa baseado nas propriedades seleccionadas;
- ▶ apoiar o “tester” com o desenvolvimento de objectivos quantitativos para teste;

Devemos parar um teste quando decidirmos que o software funciona de forma aceitável para o utilizador

critérios de adequação



critérios de adequação

- ▶ Exemplos de dados de teste (**critério de adequação**):
 - ▶ exercitar os caminhos do programa da entrada até à saída
 - ▶ executar caminhos específicos derivados de combinações de fluxos de dados, tais como definições e uso de variáveis
 - ▶ O conceito de critério de adequação é o requisito que certas características ou propriedades do código serão executadas por casos de teste
 - ▶ conduz-nos a uma abordagem chamada “análise de cobertura”, a qual na prática é usada para estabelecer objectivos de teste e avaliar dados de teste
 - ▶ No contexto de análise de cobertura os “testers” referem-se frequentemente ao critério de adequação como “critério de cobertura”

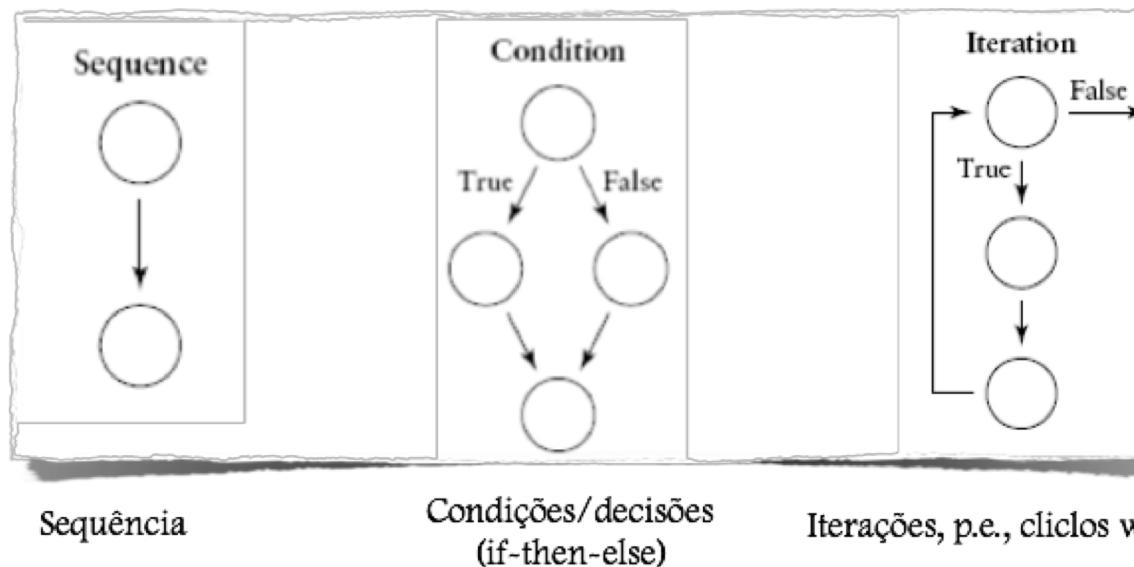
aplicação dos conceitos de cobertura

- ▶ Também apoiam os “testes de caixa preta”
- ▶ O objectivo do teste pode ser exercitar (cobrir) todos os requisitos funcionais ou todas as características do sistema.
- ▶ No entanto, contrariamente ao que acontece na abordagem de caixa preta, os objectivos de cobertura de caixa branca têm forte apoio quer teórico quer prático.

cobertura e gráficos de controlo de fluxo

- ▶ Análise de cobertura está tipicamente associada com a utilização de modelos de controlo de fluxo de dados para representar elementos estruturais e dados do programa
- ▶ Os **elementos lógicos** normalmente **considerados** para cobertura são baseados no fluxo de controlo do componente de software em teste
 - ▶ Expressões;
 - ▶ Decisões / ramos (estas influenciam o fluxo de controlo do programa);
 - ▶ Condições (expressões que avaliam verdadeiro/falso);
 - ▶ Combinações de decisões e condições;
 - ▶ Caminhos (consequências de nodos nos gráficos de fluxo)

- ▶ Os gráficos de controlo de fluxo podem ser usados pelo “tester” para avaliar o código, e desenvolver casos de teste de caixa branca
- ▶ Num programa podemos ter:



caixa branca - exemplo

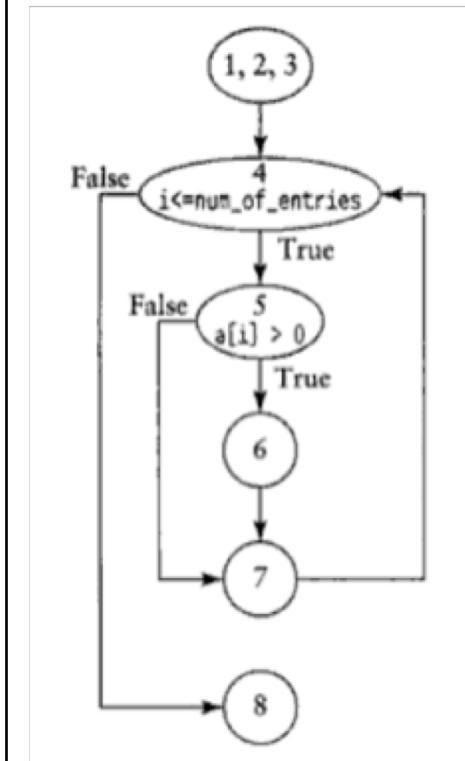
- Exemplo 1: Gráfico de controlo de fluxo (I)

```
/* pos_sum finds the sum of all positive numbers (greater than zero) stored in an integer array a. Input parameters are num_of_entries, an integer, and a, an array of integers with num_of_entries elements. The output parameter is the integer sum */
```

```
1. pos_sum(a, num_of_entries, sum)
2.   sum = 0
3.   inti = 1
4.   while (i <= num_of_entries)
5.     if a[i] > 0
6.       sum = sum + a[i]
7.     endif
8.     i = i + 1
end while
8. end pos_sum
```

testes de caixa preta

- ▶ Por questões de simplicidade, as expressões sequenciais são frequentemente omitidas ou combinadas num único bloco, que indica que se a 1ª expressão do bloco é executada, também são todas as que se seguem.
- ▶ Existem ferramentas que geram estes gráficos
- ▶ Os “testers” podem usar ferramentas para apoiar o desenvolvimento dos gráficos, especialmente para partes do código complexas



Esta representação facilita o desenho de “testes de caixa branca”

testes de caixa branca

- ▶ Vamos definir um possível caso de teste que satisfaz 100% da cobertura:
- ▶ Todas as expressões (1 a 8) são executadas no caso de teste
- ▶ **Note-se que:** podem existir diversos conjuntos de casos de teste que satisfaçam o critério....

Decision or branch	Value of variable i	Value of predicate	Test case: Value of a, num_of_entries
while	1	True	a = 1, -45,3
	4	False	num_of_entries = 3
if	2	True	
	1	False	

testes de caixa branca

- ▶ IMPORTANTE:
- ▶ Este **exemplo** de código **representa um caso especial em que foi possível alcançar toda a cobertura de ramos e expressões** com apenas um caso de teste.

Como “a” é um array foi possível atribuir valores positivos e negativos aos seus elementos, permitindo a **cobertura dos ramos “true”/“false” da expressão “if”**. Uma vez que também foram possíveis várias iterações do “while”, **conseguimos também com este caso de teste cobrir os ramos “true”/“false” do ciclo “while”**.

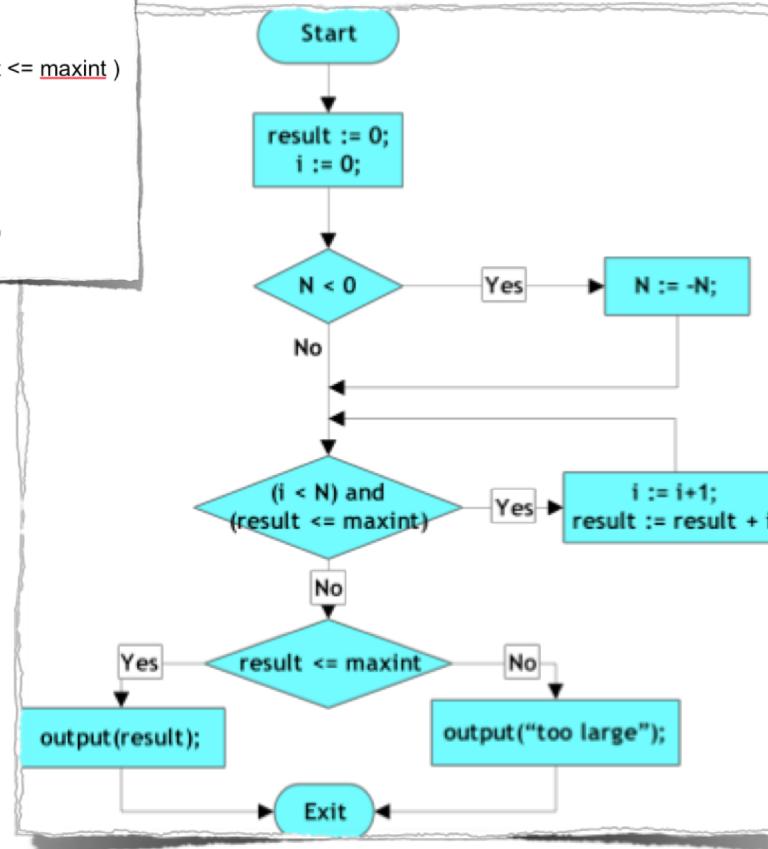
É importante reparar que o código deste exemplo não tem qualquer controlo sobre a validade dos parâmetros de entrada. Por simplicidade, assumimos que o módulo de chamada faz a validação

exemplo

```
1 PROGRAM sum ( maxint, N : INT )
2   INT result := 0 ; i := 0 ;
3   IF N < 0
4     THEN N := - N ;
5   WHILE ( i < N ) AND ( result <= maxint )
6     DO i := i + 1 ;
7       result := result + i ;
8   OD;
9   IF result <= maxint
10    THEN OUTPUT ( result )
11    ELSE OUTPUT ( "too large" )
12 END.
```

exemplo

```
1 PROGRAM sum ( maxint, N : INT )
2   INT result := 0 ; i := 0 ;
3   IF N < 0
4     THEN N := -N ;
5   WHILE ( i < N ) AND ( result <= maxint )
6     DO i := i + 1 ;
7       result := result + i ;
8   OD;
9   IF result <= maxint
10  THEN OUTPUT ( result )
11  ELSE OUTPUT ( "too large" )
12 END.
```



exemplo

- ▶ Statement coverage (cobertura de expressões)
- ▶ gerar um conjunto de casos de teste tais que cada expressão do programa é executada pelo menos uma vez
- ▶ Critério de caixa branca mais fraco
- ▶ Note-se que: em caso de expressões inacessíveis, a sua cobertura não é possível
- ▶ Teste para **cobertura completa de expressões:** →

inputs		outputs
maxint	N	result
10	-1	1
0	-1	too large

exemplo

- ▶ Decision (or Branch) coverage (Cobertura de decisões (ou ramos))
- ▶ Executar todos os ramos de um programa: cada resultado possível de cada decisão ocorre pelo menos uma vez
- ▶ Exemplo:
- ▶ Decisão simples: IF b THEN s1 ELSE s2
- ▶ Decisão múltipla:
 - ▶ CASE x OF
 - ▶ 1:
 - ▶ 2:
 - ▶ 3:
- ▶ Mais forte que a cobertura de expressões
- ▶ IF THEN sem ELSE – se a condição é sempre verdadeira todas as expressões são executadas, mas a cobertura de ramos não é alcançada

exemplo

- ▶ Decision (or Branch) coverage (Cobertura de decisões (ou ramos))



inputs		outputs
maxint	N	result
10	-1	1
0	-1	too large

Este teste de cobertura completa de expressões não é suficiente para cobertura de decisões (ramos)!!!!

exemplo

- Decision (or Branch) coverage (Cobertura de decisões (ou ramos))



Considerar:

inputs	outputs	
maxint	N	result
10	3	6
0	-1	too large

Para cobertura completa de decisão (ramos)

exemplo

- ▶ Condition Coverage (Cobertura de Condições)
- ▶ desenhar casos de teste de tal modo que **cada resultado possível de cada condição, numa condição composta ocorre pelo menos uma vez**
- ▶ Exemplo:
 - ▶ decisão ($i < N$) AND ($result \leq maxint$)
 - ▶ Consiste em duas condições: ($i < N$) e ($result \leq maxint$)
 - ▶ Devem ser desenhados casos de teste de tal modo que **cada condição assume o valor verdadeiro e falso pelo menos uma vez**
 - ▶ Note-se que os últimos casos de teste dos slides anteriores já garantem cobertura de condições (e ramos).

exemplo

- ▶ Independent path coverage (Cobertura de caminhos independentes)
 - ▶ gerar um caso de teste para cada caminho independente
 - ▶ O número de caminhos linearmente independentes é dado pela McCabe's cyclomatic complexity do programa $V(G)$

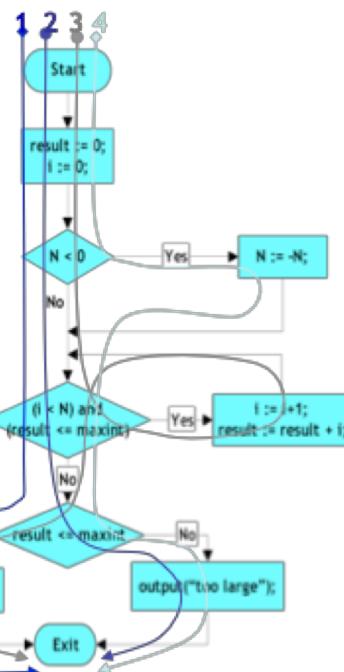
$$V(G) = E - N + 2$$

E = number of edges

N = number of nodes

exemplo

- ▶ Independent path coverage (Cobertura de caminhos independentes)



Número de caminhos independentes

$$= 12 - 10 + 2 = 4$$

Casos de teste

Path	inputs		outputs
	maxint	N	result
1	1	0	0
2	-1	0	too large
3	-1	-1	too large
4	10	1	1

testes de caixa branca

- ▶ Path coverage (Cobertura de caminhos)
 - ▶ executar todos os caminhos possíveis de um programa
 - ▶ critério de caixa-branca forte (baseado na análise de controlo de fluxo)
- ▶ Normalmente impossíveis: infinidade de caminhos (em caso de ciclos)
- ▶ Portanto, não são uma opção realística

boas práticas

- ▶ Testes de caixa branca: como aplicar?
 - ▶ não começar com desenho de testes de caixa branca!!!!
 - ▶ **começar com casos de teste de caixa preta**
 - ▶ garantir cobertura de caixa branca (expressões, ramos, condições,)
 - ▶ **usar uma ferramenta de cobertura**
 - ▶ desenhar os testes de caixa branca adicionais para código não coberto

Referências Bibliográficas

- ▶ Burnstein, I., Practical Software Testing: A Process-Oriented Approach. Springer Professional Computing, 2003. – capítulos 4 e 5
- ▶ PRESSMAN, Roger S., Software Engineering - A practitioner's Approach. McGraw-Hill, 5a. Edição, 852pp.