Testes de *software* - Teste funcional

Vitor Alcântara de Almeida

Universidade Federal do Rio Grande do Norte
Natal, Brasil

30 de outubro de 2014

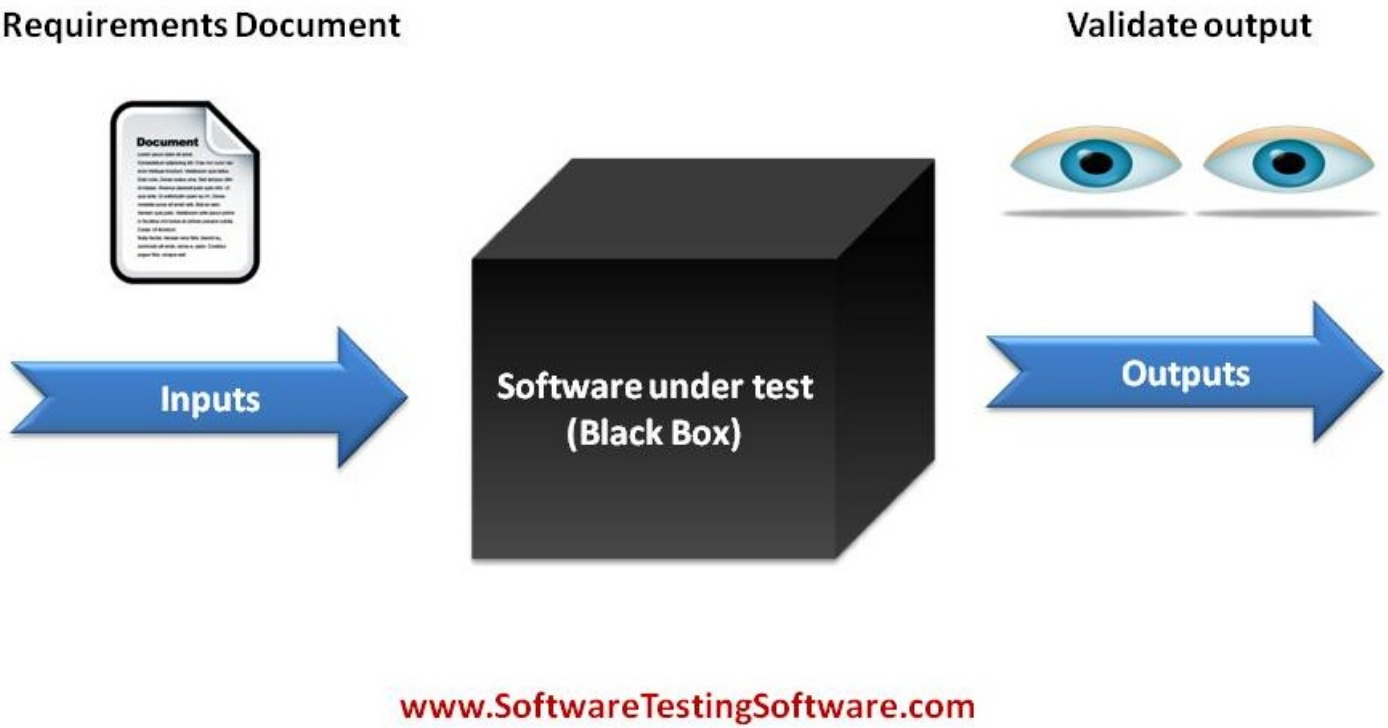
Sumário

Objetivo

- Definição de testes funcionais
- Tipos de critérios existentes
- Critérios de partição:
  - Classes de equivalência
  - Valor limite
  - Teste funcional sistemático

O que é teste funcional?

- Técnica de teste no qual o sistema é considerada uma caixa preta
- São avaliadas as saídas produzidas pelo sistema de acordo com as entradas fornecidas



- Detalhes de implementação não são consideradas nos testes

Que valores escolher para os testes?

- Uma forma de testar é através da verificação de todas as entradas possíveis - teste exaustivo

Programa exemplo

- Porém, seja abaixo a especificação de um programa

Cadeia de Caracteres

1. O programa solicita do usuário:
   1. Um inteiro positivo T no intervalo entre 1 e 20
   2. Uma cadeia de caracteres CC com tamanho T
   3. Um caractere C
2. Programa retorna posição onde se encontra C em CC
3. O programa pode procurar por um caractere mais de uma vez (interativamente)

Testes possíveis

- Quantas combinações de entradas podem ser testadas?
- Inviável testar para todas as entradas

Subconjunto do domínio de entrada

- É impossível testar todos os elementos do domíno ==> definir um subconjunto
- Mas, qual critério utilizar para selecionar o subconjunto?

Possíveis soluções

Random testing

Usar valores quaisquer - aleatórios

- Abordagem possível
- Mas há outras técnicas mais padronizadas para encontrar *bugs*

Critérios de teste estrutural

- Podemos definir critérios de teste com base no conhecimento da estrutura interna da implementação
- Ex.:
  - Todos os métodos foram executados?
  - Todas as linhas de comando foram executadas?
  - Todos os nós de decisão foram executados?
- Critérios de caixa branca - não é o propósito desta aula

Critérios de teste funcional

- Também podemos definir critérios de teste com base no conhecimento dos requisitos do programa
- ...independemente da estrutura interna da implementação
- Ex.:
  - Os valores limites de cada campo foram exercitados?
  - Os valores mais utilizados foram exercitados?
- Veremos nesta aula alguns critérios funcionais

Alguns critérios de teste funcional

- Como podemos criar testes de forma padronizada a partir da especificação do sistema?
- Alguns critérios foram propostos neste sentido:
  - Particionamento em classes de equivalência
  - Análise do valor limite
  - Teste funcional sistemático

Sumário

Classe de Equivalência

- O domínio de entrada do *software* é dividido em classes (ou partições)
- Cada classe de equivalência é um subdomínio de entrada no qual os valores de entrada produzem comportamento similar

Exemplo a partir de “Cadeia de Caracteres´´

- O tamanho da cadeia x deve ser entre 1 e 20
- Uma classe para $1 \leq x \leq 20$
- Uma classe para $x < 1$ e uma classe para $x > 20$

Classe de Equivalência

- A base deste critério é que todos os dados de uma mesma partição têm a capacidade de revelar as mesmas falhas
- Se o sistema falhar com um valor de uma partição, deverá falhar com qualquer outro valor desta mesma partição
  - O contrário também é válido

Classe de Equivalência

- Todos os valores do domínio de entrada devem pertencer a uma classe
- Para identificar valores com comportamento similar, analisamos conjuntos de **entrada** e de **saída**
  - Como criar as classes a partir desta análise?

Tipos de Classes de Equivalência

- As partições são então classificadas em dois tipos:
  - **Classes válidas:** valores válidos no domínio de entrada
  - **Classes inválidas:** valores inválidos no domínio de entrada

Diretrizes para criação de classes de equivalência

Variável de entrada	Classes de equivalência
Intervalo de valores	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
Quantidade de valores	Uma classe válida para número de valores igual ao previsto
	Uma classe inválida para número de valores maior ou menor que o número previsto
Valores determinados e distintos	Uma classe válida para cada valor determinado
	Uma classe inválida para qualquer outro valor diferente

Como criar testes de classes de equivalência

- Para gerar testes de acordo com este critério, dois passos devem ser executados:
  1. identificação das classes de equivalência
  2. gerar casos de teste selecionando **pelo menos** um elemento de cada classe
     - ⋆ Deste modo, o número de casos de teste é menor e pode encontrar mais erros

Programa exemplo

- Relembrando o programa exemplo:

Cadeia de Caracteres

1. O programa solicita do usuário:
   1. Um inteiro positivo T no intervalo entre 1 e 20
   2. Uma cadeia de caracteres CC com tamanho T
   3. Um caractere C
2. Programa retorna posição onde se encontra C em CC
3. O programa pode procurar por um caractere mais de uma vez (interativamente)

Exemplo de aplicação de Classes de equivalência

- Para o programa especificado temos quatro entradas:
  - T - tamanho da cadeira de caracteres
  - CC - uma cadeia de caracteres
  - C - um caractere a ser procurado
  - O - a opção por procurar mais caracteres
- Qualquer valor de CC e C é válido no sistema
  - Não é necessário definir classes de equivalência
- $1 <= T <= 20$
- $O = TRUE$ ou $O = FALSE$

Exemplo de aplicação de Classes de equivalência

- Considerando o domínio de saída, temos duas alternativas:
  - O sistema retorna a posição onde o caractere foi encontrado na cadeia de caracteres
  - O sistema retorna uma mensagem dizendo que o caractere não foi encontrado
- Com estas informações, nova partição pode ser feita no domínio de entrada:
  1. $C \in CC$
  2. $C \notin CC$

Exemplo de aplicação de Classes de equivalência

- Ao total, 6 classes de equivalência, mostradas abaixo:

Variável de entrada	Classes de equivalência válidas	Classes de equivalência inválidas
T	$1 <= T <= 20$ (V1)	$T < 1$ (I1a) e $T > 20$ (I1b)
O	Sim (V2)	Não (I2)
C	Caractere que pertence à cadeia (V3)	Caractere que não pertence à cadeia (I3)

- Deve ser criado um caso de teste para cada classe de equivalência

Testes criados

Variáveis de entrada				Saída esperada	Classe
T	CC	C	O		
0				’entre com um inteiro entre 1 e 20’	I1a
34				’entre com um inteiro entre 1 e 20’	I1b
3	abc	c		’o caractere c aparece na posição 3 da cadeia’	V1 & V3
			s		V2
		k		’o caractere k não partence à cadeia’	I3
			n		I2

Avaliação do critério

- Redução do domínio de entrada
- Testes baseados unicamente na especificação
- Mais adequado para aplicações onde variáveis de entrada são identificáveis com facilidade e assumem valores específicos
- Não determina combinações de testes
  - Poderia cobrir as classes de equivalência de maneira mais eficiente

Sumário

Valor limite

- Estudos mostram que casos de testes que exploram condições limites têm uma maior probabilidade de encontrar defeitos (MEYERS, 2004)
- Dada uma classe de equivalência e um valor no limite da mesma, tais condições correspondem a testar:
  - O valor limite da classe
  - O valor imediatamente acima do limite da classe OU imediatamente abaixo do limite da classe

Exemplo a partir de “Cadeia de Caracteres´´

- Para a mesma cadeia x que deve ser entre 1 e 20
- Realizar testes para $0, 1, 20, 21$

Valor limite

- Critério utilizado conjuntamente com classes de equivalência
  - São escolhidos valores limites ao invés de valores aleatórios
- Algumas diretrizes foram estabelecidas para a criação dos testes

Diretriz 1: Faixa de valores

- Definir testes para:
  1. Os limites do intervalo
  2. Os valores subseq. ao intervalo se eles exploram classes inválidas

Exemplo?

Para $x \in \mathbb{N} \wedge -10 <= x <= 10$ testar os valores: **-11;-10;10;11**

Diretriz 2: Quantidade de valores

- Aplica-se a mesma diretriz se parâmetro for uma quantidade de valores
- Ex: Se uma entrada determina serem inseridos de 1 a 255 valores, inserir testes para 0,1,255 e 256 valores

Mais diretrizes

- Usar as diretrizes 1 (Faixa de valores) e 2 (Quantidade de valores) para verificar também as condições de saída
- Se entrada ou saída for um conjunto ordenado, dar maior atenção aos primeiro e último elementos desse conjunto
- Usar a intuição para definir outras condições limites!

Classes de equivalência do exemplo

Variável de entrada	Classes de equivalência válidas	Classes de equivalência inválidas
T	$1 <= T <= 20$ (V1)	$T < 1$ (I1a) e $T > 20$ (I1b)
O	Sim (V2)	Não (I2)
C	Caractere que pertence à cadeia (V3)	Caractere que não pertence à cadeia (I3)

Novos casos de teste

Entrada				Saída esperada
T	CC	C	O	
21				’entre com um inteiro entre 1 e 20’
0				’entre com um inteiro entre 1 e 20’
1	a	a		’o caractere a aparece na posição 1 da cadeia’
			s	
		x		’o caractere x não pertence à cadeia
			n	
20	abcdefghijkl mnopqrst	a		’o caractere a aparece na posição 1 da cadeia’
			s	
		t		’o caractere t aparece na posição 20 da cadeia
			n	

Avaliação do critério

- Vantagens e desvantagens similares ao critério Particionamento de Equivalência
- Mas, neste caso, há diretrizes para aproveitar melhor os testes

Sumário

Teste funcional sistemático

- Combina Particionamento de Equivalência com Análise do Valor Limite
- Mínimo de dois testes para cada partição (classe de equivalência)
  - Minimiza que defeitos coincidentes mascarem falhas

Exemplo

- Um programa retorna o quadrado de um número. Se valor de entrada for 2, valor produzido será 4
- Qual operação realizada? 2*2 ou 2+2?

- Possui critérios mais rigorosos para cada tipo de entrada

Diretrizes para testes funcionais sistemáticos

1. Valores numéricos - testar os extremos e um valor dentro do intervalo
2. Valores discretos - testar cada valor discreto e um valor fora dos já determinados
3. Testar tipos diferentes de dados e casos especiais

Exemplos:

1. Limites de representação (-32768 e 32767 para int16)
2. Inserir valor de tipo diferente do previsto (int no lugar de char)

Mais diretrizes para testes funcionais sistemáticos

1. Números reais - verificar limites igualmente a números inteiros
   1. Comparações podem não ser exatas - definir margem de erro
2. Intervalos variáveis - o intervalo de uma variável depende do valor de outra variável
   1. Testar combinações dos possíveis valores

Exemplo: $0 \leq x \leq y$

1. Entradas válidas: $x = y = 0$, $x = 0 < y$, $0 < x = y$, $0 < x < y$
2. Entradas inválidas: $y = 0 < x$, $0 < y < x$, $x < 0$, $y < 0$

Diretrizes para Matrizes

1. Testar os elementos do arranjo como se fossem variáveis comuns
2. Devemos testar o arranjo com o tamanho mínimo, intermediário e máximo
3. Podemos testar linhas e colunas em separado
    1. Podemos testar Matriz como:
        1. Uma única estrutura
        2. Uma coleção de subestruturas testadas independemente

Diretrizes para *Strings* e Entradas de texto

1. Explorar *string* com comprimentos variáveis
2. Validar os caracteres que os compõem
    1. *strings* podem conter somente caracteres alfabéticos, alfanuméricos ou também possuir caracteres especiais

Sobre o critério de teste

- As diretrizes não precisam ser seguidas sempre à risca
- Porém, quanto mais testes, maiores as chances de evitar erros no sistema

Exemplo de aplicação

- Os casos de testes já produzidos são válidos
- Mas, outros casos de testes precisam ser feitos pra atender este critério

Casos de teste funcional sistemático

Entrada				Saída esperada
T	CC	C	O	
a				entre com um inteiro entre 1 e 20
1.0				entre com um inteiro entre 1 e 20
1	!	’ ’	n	o caractere ’ ’ não pertence à cadeia
1	}		n	o caractere ’ ’ não pertence à cadeia
20	¡‘#$%&()*+´ /01234567	!	s	o caractere ! aparece na posição 1 da cadeia
		’	s	o caractere ’ aparece na posição 2 da cadeia
		+	s	o caractere + aparece na posição 10 da cadeia
		6	s	o caractere 6 aparece na posição 19 da cadeia
		7	n	o caracatere 7 aparece na posição 20 da cadeia
2	ab	b	nao	o caracatere b aparece na posição 2 da cadeia
3	a2b	2	0	o caracatere 2 aparece na posição 2 da cadeia

Avaliaçao do critério

- Apresenta os mesmos problemas dos critérios de Particionamento de Equivalência e Análise do Valor Limite
- Mas, assim como Análise do Valor Limite, este fornece diretrizes para geração de testes
- Enfatiza mais de um caso de teste por partição e/ou limite
- Melhor cobertura do código do produto testado

Sumário

Ferramentas de testes funcionais

- Existem uma série de ferramentas de testes em C, como:
  - Cunit
  - Quick-Check
  - API Sanity Checker
  - Automated Test Framework
  - Parasoft C/C++ test
  - Check
  - Outros...
- Introduzo nesta aula o **Cunit** para realizar testes funcionais

Cunit

- *Framework* leve de testes unitários para C
- Biblioteca estática *linkado* ao código do usuário
- Permite testes automáticos e interativos

Exemplo de uso

- Seja abaixo um código em C:

  insertion_sort.c

- Um caso de teste simples pode ser construído:

  insertion_sort_teste.c

Execução do teste

- Compilação:

```
gcc insertion_sort.c insertion_sort_teste.c
                -lcunit -o teste
```

- Execução:

```
./teste
```

Sumário

Referências