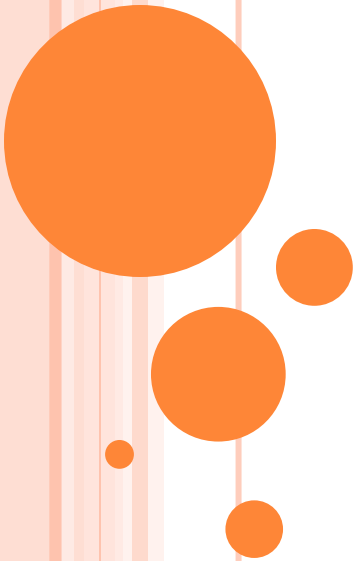


# Testes de Caixa Preta

Paulyne Jucá (paulyne@ufc.br)

# Definição

- Um estratégia de teste que não leva em consideração o funcionamento interno do componente ou sistema
  - Mesmo que tenha acesso ao código, este não é usado

- Também chamada de:
  - Teste funcional
  - Teste de caixa fechada
  - Teste Opaco


- Estratégia mais usada por testadores

# Objetivo

- Verificar se o software desenvolvido atende aos requisitos especificados
  - Entradas válidas são aceitas e geram as saídas esperadas e entradas inválidas geram as mensagens de erro esperadas
  - Encontrar inclusive coisas que não foram implementadas
- Verificar se o software desenvolvido atende às expectativas do usuário

- Exemplos e erros encontrados:
  - Funções incorretas ou não implementadas
  - Erros na interface
  - Erro nas estrutura de dados
  - Erro no acesso à base de dados
  - Falhas de comportamento ou desempenho
  - Falhas na iniciação ou término da execução

- Geralmente fazem parte da fase de teste, mas podem ser usados em todas as fases

# Vantagens

- Teste são feitos do ponto de vista do usuário e irão ajudar a identificar diferenças entre implementação e especificações
- Não necessita de conhecimento da linguagem de programação usada
  - Será?
- Pode ser feito por equipe independente
  - Desenvolvedores tem pena de quebrar seus códigos
- Os casos de teste podem ser planejados assim que as especificações são feitas

# Desvantagens

- Alguns caminhos de execução podem não ser testados
- Requisitos não claros ou mal escritos afetam a qualidade dos casos de teste
  - Como saber o que/como testar, se o requisito não está claro?
- A escolha dos dados de entrada dos casos de teste influencia a qualidade do teste. Bons dados de entrada dependem da experiência do testador.
  - Valores randômicos ou valores planejados?
    - Resposta: combinação de ambos

# Caixa Preta [7]

- Os passos básicos para se aplicar um critério de teste caixa preta são os seguintes:
  - A especificação de requisitos é analisada.
  - Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  - As saídas esperadas para as entradas escolhidas são determinadas.
  - Os casos de testes são construídos.
  - O conjunto de teste é executado.
  - As saídas obtidas são comparadas com as saídas esperadas.
  - Um relatório é gerado para avaliar o resultado dos testes.

# Como escolher os valores de entrada para os testes?

# Como escolher os valores de entrada para os testes?

**Teste exaustivo (que testa todas as entradas possíveis) pode ser impossível de fazer.**

**Qual a opção?**

# Escolha Aleatória de valores

- Funcionalidade a ser testada: “O programa deve receber uma sequencia de caracteres e quebrar em linhas de até 60 caracteres”

- Qual o primeiro teste que faríamos?

# Escolha Aleatória de valores

- 1° teste: dslkngiurewht
- 2° teste: dsjntriuwtoirmgvgjbryg32tr5iu45nyhby
- 3° teste: fjewbt ewtbwt wetbwtv

- Risco? Polarizar os testes com sequencias com menos de 60 caracteres

- O que deixamos de fora? (casos especiais fortemente baseados na experiência)
  - Testamos sequencias com mais de 60 caracteres (com e sem espaço)? Como o programa respondeu nos dois casos?
  - Testamos uma sequencia de mais de 60 espaços?
  - Testamos sequencias de caracteres especiais?
  - Testamos strings nulas?
  - Testamos maiúsculas e minúsculas? Números? Acentos? Pontuações?
  - Testamos só com “enter”? Tab + “enter”?
  - Testamos apenas sequencias que começam com letra?

# Escolha Aleatória de valores

- Vantagem
  - Simples
  - Liberdade
- Desvantagem
  - Pode nunca encontrar erros

- Significa que é ruim?
  - Não!!! Especialmente quando o testador identifica "casos especiais" na descrição da funcionalidade
    - Ex: "The quick brown fox jumps over the lazy dog"
  - Podemos criar geradores de entradas aleatórias para ajudar no teste. ☺

# Dá para confiar só em testes randômicos?

**Mas o que mais podemos fazer?**

# Abordagens para escolha de valores

- Abordagens:
  - Particionamento de Equivalências
  - Análise de Valor Limite
  - Combinação de Pares
  - Baseado em Catálogos
  - Teste Funcional Sistemático
  - Error-Guessing

# Abordagens para escolha de valores

- Selecionando a abordagem:
  - Natureza e forma da especificação:
    - Ex: “para compras acima de R$ 100, o parcelamento pode ser em até 3x” sugere uma abordagem de particionamento de categorias
  - Experiência dos projetistas de testes e da organização:
    - Escolher técnicas que dominem melhor quando mais de uma for aplicável
  - Ferramentas:
    - Algumas técnicas exigem ferramentas específicas que podem custar caro ou não estar disponíveis

# Abordagens para escolha de valores

- Selecionando a abordagem:
  - Restrições de orçamento e qualidade:
    - Ex: se precisamos de testes rápidos e automatizados e não temos requisitos de confiabilidade altos podemos usar geração randômica
    - Custos de treinamento de equipe influenciam a decisão
    - Tamanhos da suíte de teste influencia a decisão
  - Custos do código de suporte
    - Automatizar os casos de teste custam $, mas valem se forem usados muitas vezes
    - Se os casos de teste forem executados manualmente, talvez seja melhor optar por uma suíte de teste menor

- Ideal: encontrar o balanceamento correto para uso de cada abordagem para satisfazer a necessidade de teste dentro do custo, tempo e qualidade desejadas.

# Particionamento de Equivalências [8]

- Sistema de recursos humanos – empregar pessoas com base na idade (Copeland, 2004).

| Idade | Emprego |
|---|---|
| 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?
  - Testar todos os valores de 0 a 99?

# Particionamento de Equivalências [8]

- Encontrar classes (intervalos) de valores que tem comportamento equivalente
- Encontrar intervalos onde todos os valores representem a mesma possibilidade (bastando testar um deles)
  - Se funciona para um, funciona da mesma maneira para os outros
- Esses intervalos são chamados de classes de equivalência
  - 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
- Essa abordagem assume que é possível identificar esses intervalos na especificação do sistema

# Particionamento de Equivalências [8]

- Sistema de recursos humanos – empregar pessoas com base na idade (Copeland, 2004).

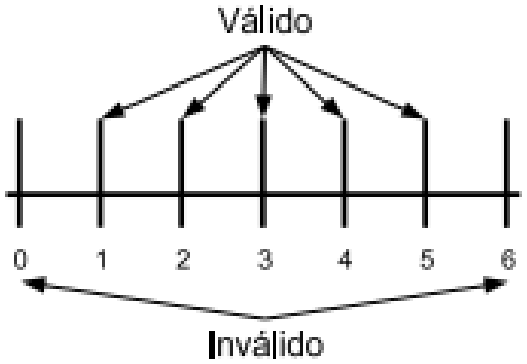
| Idade | Emprego |
|---|---|
| 0 – 16 (<= 15) | Não empregar |
| 16 – 18 (<=17) | Pode ser empregado tempo parcial |
| 18 – 55 (<=54) | Pode ser empregado tempo integral |
| 55 - 99 | Não empregar |

- Para o caso acima foram identificadas 4 classes de equivalência
  - 4 casos de teste em vez dos 100 do teste exaustivo
  - + os testes de casos inválidos (negativos, por exemplo)

# Particionamento de Equivalências [8]

- Diferentes tipos de dados exigem diferentes classes de equivalência
  - Intervalos contínuos


  - Intervalos discretos
    - Ex: números inteiros
    - Válida: 2
    - Inválidas -1 e 8

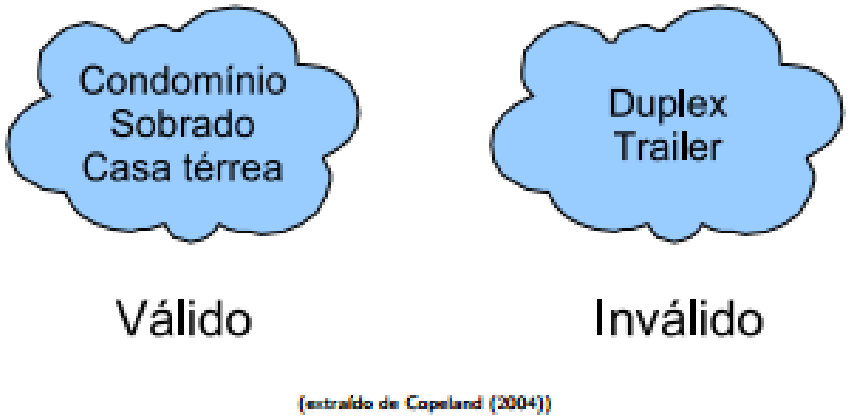



(extraído de Copeland (2004))

- Em geral são definidas 2 classes não válidas e 1 válida

# Particionamento de Equivalências [8]

- Diferentes tipos de dados exigem diferentes classes de equivalência (continuação)
  - Intervalos de valores de múltipla escolha




(extraído de Copeland (2004))

  - Válido: pode se escolher um ou mais valores dentro das possibilidades de valor válido
  - Inválido: pode se escolher um ou mais valores dentro das possibilidades de valor inválido
  - Pode ser difícil criar um caso de teste para cada classe válida.

# Particionamento de Equivalências [8]

- Diferentes tipos de dados exigem diferentes classes de equivalência (continuação)
  - Intervalos de valores de múltipla escolha
    - Pode ser difícil criar um caso de teste para cada classe válida.
      - Criar o menor número de casos de teste que cubra 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 |

# Particionamento de Equivalências [8]

- “O programa deve determinar 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”
  - Ex válidos: a, a123, abcd2
  - Ex inválidos: 1abc, cont*1, abcdefg
  - Quais as classes de equivalência?

# Particionamento de Equivalências [8]

- Classes de Equivalências

| 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)$\}$

# Particionamento de Equivalências [8]

- “O programa solicita ao usuário um inteiro positivo entre 1 e 20 e então pede uma cadeia de caracteres com o tamanho do inteiro fornecido anteriormente. Após isso, o programa solicita um caractere e retorna a posição da primeira ocorrência do caractere na cadeira ou um mensagem indicando que o caractere não está presente na cadeia.”
  - Quais as classes de equivalência válida e inválidas?
  - Dê exemplos de casos de teste para esse programa.

# Particionamento de Equivalências

- Classes de equivalência

| Variável de entrada | Classe de equivalência válida | Classe de equivalência inválida |
|---|---|---|
| Tamanho da cadeia (T) | 1 <= T <= 20 | T < 1 e T > 20 |
| Caractere procurado (C) | Pertence | Não pertence |

# Particionamento de Equivalências

- Casos de Teste

| Valores de Entrada | | | Saída esperada |
|---|---|---|---|
| Tamanho | Cadeia | Caractere | |
| 34 | | | Entre com um inteiro entre 1 e 20 |
| 0 | | | Entre com um inteiro entre 1 e 20 |
| 3 | abc | c | Posição 3 |
| 3 | def | k | Não pertence |

# Valor Limite [8]

- Toma como base as classes de equivalência
- Premissa: Condições limite das classes de equivalência tem mais chance de dar erro
  - Testar limite -1
  - Testar limite
  - Testar limite +1

- Ex1: para uma variável que aceita valores entre -1.0 e +1.0 os casos de teste teriam como entrada os valores – 1.001; -1.0 ; – 0.999 ; + 0.999; + 1.0 e + 1.001
- Também pode ser aplicada em valores de saída

# Valor Limite [8]

- Para o exemplo anterior....

| Idade | Emprego |
|---|---|
| 0 – 16 (<= 15) | Não empregar |
| 16 – 18 (<=17) | Pode ser empregado tempo parcial |
| 18 – 55 (<=54) | Pode ser empregado tempo integral |
| 55 - 99 | Não empregar |

- Os casos de teste incluiriam as entradas -1, 0, 1, 14, 15, 16, 17, 18, 19, 53, 54, 55, 56, 98, 99, 100.

# Valor Limite

- Classes de equivalência do exemplo anterior

| Variável de entrada | Classe de equivalência válida | Classe de equivalência inválida |
|---|---|---|
| Tamanho da cadeia (T) | 1 <= T <= 20 | T < 1 e T > 20 |
| Caractere procurado (C) | Pertence | Não pertence |

# Valor Limite

- Casos de Teste no Valor Limite

| Valores de Entrada | | | Saída esperada |
|---|---|---|---|
| Tamanho | Cadeia | Caractere | |
| 21 | | | Entre com um inteiro entre 1 e 20 |
| 0 | | | Entre com um inteiro entre 1 e 20 |
| 1 | a | a | Posição 1 |
| 1 | a | x | Não pertence |
| 20 | abcjtahenso qujkmkeux | c | Posição 3 |
| 20 | abcjtahenso qujkmkeux | u | Posição 13 |
| 19 | abcjtahenso qujkmkeu | z | Não pertence |

# Referências

- Os livros de referência da disciplina +


1. http://www.testinggeek.com/blackbox-testing
2. http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf
3. http://www.buzzle.com/editorials/4-10-2005-68349.asp
4. http://www.codeproject.com/Articles/37078/Black-box-Testing-Techniques
5. http://softwaretestingfundamentals.com/black-box-testing/
6. http://www.softwaretestinghelp.com/black-box-testing/
7. http://www.inf.ufg.br/~auri/curso/arquivos/funcional01.pdf
8. http://www.inf.ufg.br/~auri/curso/arquivos/funcional02.pdf