Aula 19
Testes 1

Bruno Cafeo e Alessandro Garcia

OPUS Group/LES/DI/PUC-Rio

Maio 2016

Slides adaptados de: Staa, A.v. Notas de Aula em Programacao Modular; 2008.

Especificação

- Objetivo dessa aula
  - Apresentar os conceitos básicos de teste
  - Discutir o processo e as principais atividades realizadas ao se testar um sistema
  - Citar as técnicas de teste que serão estudadas no curso
  - Apresentar o teste caixa-preta (teste funcional)

- Referência básica:
  - Capítulo 15

  - Slides adaptados de: Staa, A.v. Notas de Aula em Programacao Modular; 2008.

Sumário

- Motivação
- Qualidade de Software
- O que são testes?
- Objetivos dos testes
- Processo e atividades de teste
  - Diagnose, Depuração e Registro de Falhas
- Técnicas e critérios de teste
  - Teste Caixa-Preta

Motivação – "Trem fantasma"

- Na década de 90, na Inglaterra, ocorreu um problema nos sensores das linhas férreas situadas em túneis
- O sistema alertava o trem a parar devido à suspeita de outro trem na linha
- A causa do problema era a névoa de água salgada que "confundia" os sensores
- Motivo da falha:
  - Provavelmente os requisitos do sistema não levavam em conta a névoa de água salgada como um possível evento

Motivação – Metrô em Nova Iorque

- Em 1995, duas composições colidiram matando um maquinista e ferindo 54 pessoas
- A distância entre os sinais (projetado em 1918) era menor que a distância de parada necessária para os trens em 1995 (maiores, mais pesados e mais rápidos)
- Os trens foram atualizados sem modificação no sistema de controle
- Motivo da falha:
  - Atualização de uma parte do sistema sem validar o sistema como um todo.

Motivação – Ariane 5

- Em 1996, o veículo espacial Ariane 5 saiu de sua rota e explodiu segundos após o lançamento. Levou uma década para ser desenvolvido e custou 7 bilhões de dólares

- Componentes reutilizados do veículo Ariane 4 foram a causa do acidente

- Motivo da falha:
  - A variável que armazenava cálculo da velocidade horizontal do foguete deveria ter 64 bits (floating point), mas possuía 16 bits (signed integer). O valor era maior que 32.767 gerando uma falha de conversão!!!

Motivação

- Alguém já testou algum produto de software?
- Quais foram os maiores desafios?

Motivação

- Problemas comumente citados:
  - Não há tempo suficiente para o teste
  - Inúmeras combinações de entrada para serem exercitadas
  - Dificuldade em determinar os resultados esperados
  - Requisitos do software incompletos ou que mudam rapidamente
  - Não há treinamento para gerentes e desenvolvedores
  - Dificuldade em encontrar ferramentas de apoio
  - ...

Qualidade do software

- Por que é difícil construir software com qualidade?
- Existe algum mecanismo para garantir a qualidade do software? Como utilizá-lo eficientemente?

Qualidade do software

- Por que é difícil construir software com qualidade?
- E qualidade do s

O que é qualidade?

É a totalidade de características e critérios de um produto ou serviço que exercem suas habilidades para satisfazer às necessidades declaradas ou envolvidas.

Qualidade do software

- Por que é difícil construir software com qualidade?
- Existe algum mecanismo para garantir a qualidade do software? Como utilizá-lo eficientemente?
- **Garantia de qualidade**: Conjunto de atividades aplicadas durante todo o processo de desenvolvimento que visam a garantir que tanto o processo de desenvolvimento quanto o produto de software atinjam os níveis de qualidade especificados

Garantia de Qualidade

- Nenhuma técnica de garantia da qualidade assegura que um artefato nunca gerará erros
  - medição
  - revisão, inspeção
  - argumentação, prova formal
  - instrumentação
  - *teste*

- Logo devemos utilizar uma variedade de técnicas a começar com o desenvolvimento correto por construção

O que são testes?

- Análise dinâmica do produto de software
  - Processo de executar o software de modo controlado, observando seu comportamento em relação aos requisitos especificados
- Processo de executar um programa com a intenção de encontrar erros
  - O teste bem sucedido é aquele que consegue determinar casos de teste que resultem na falha do programa sendo analisado.
- Basicamente: Entrada → Processamento → Saída
  - Comparação entre saída esperada e saída obtida

O que são testes?

- Testes são técnicas de controle da qualidade baseados na condução de experimentos controlados
- Em um experimento controlado:
  - dispõe-se de uma hipótese: o módulo a testar corresponde exatamente à sua especificação
    - não tem funcionalidade a mais
    - não tem funcionalidade a menos
    - satisfaz todos os requisitos não funcionais
  - formula-se um conjunto de experimentos:
    - a massa de teste
      - conjunto de dados e comandos
    - os resultados dos testes que suportam a hipótese
      - os resultados esperados para cada dado e/ou comando
    - procura-se identificar condições que tenham elevada chance de mostrar que a hipótese não vale

O que são testes?

- Em um experimento controlado (cont.):
  - executa-se o teste (efetua-se o experimento)
    - obtêm-se os resultados dos testes
    - comparam-se os resultados esperados com os obtidos
  - caso todos os experimentos comprovem que o resultado esperado é igual ao obtido (dentro de uma tolerância), conclui-se que o programa corresponde à sua especificação, considerando o teste realizado
    - a conclusão será sempre dependente da massa de teste utilizada
      - a rigor testes deveriam ser repetidos com variadas massas de teste e variadas condições de realização deles
    - quanto maior o rigor do experimento, mais confiança poder-se-á depositar na conclusão

Por que é difícil garantir a qualidade (teste)?

- Testes são
  - caros
    - há quem diga que testes correspondem a entre 30 e 50% do esforço de desenvolvimento
  - demorados
    - por meio da automação dos testes reduz-se significativamente (ordens de grandeza) o esforço despendido
  - ineficientes
    - Se aplicado de forma não sistemática, encontram poucas falhas a cada execução
  - ineficazes
    - estatísticas mostram que somente encontram cerca de 65% dos problemas
  - devem ser projetados cuidadosamente
    - visando aumentar eficiência e eficácia

Objetivos do teste

- Em geral o teste deve ser orientado à destruição
  - procurar demonstrar que o módulo não corresponde exatamente à especificação
    - procurar demonstrar que o módulo “não funciona”
  - ao invés de demonstrar que corresponde à especificação
    - ao invés de “ver se funciona”
  - algumas vezes o teste será orientado à avaliação (auditoria funcional)
    - verificar se resolve o problema do usuário
    - quão bem o faz
    - mesmo aqui o interesse é descobrir os casos em que não o faz tão bem quanto o esperado

Objetivos do teste

- Encontrar o maior número de falhas relevantes possível
  - encontrar todas é um ideal em geral não alcançável
- Encontrar falhas funcionais
  - o módulo não corresponde exatamente à sua especificação
- Encontrar falhas não funcionais
  - utilizabilidade
  - desempenho
  - tempo de resposta
  - capacidade
  - ...
- Estimar o grau de qualidade do teste
  - completeza (cobertura) do teste
    - ver contadores de passagem, aula 23
  - riscos de defeitos remanescentes
    - os defeitos remanescentes são sempre desconhecidos

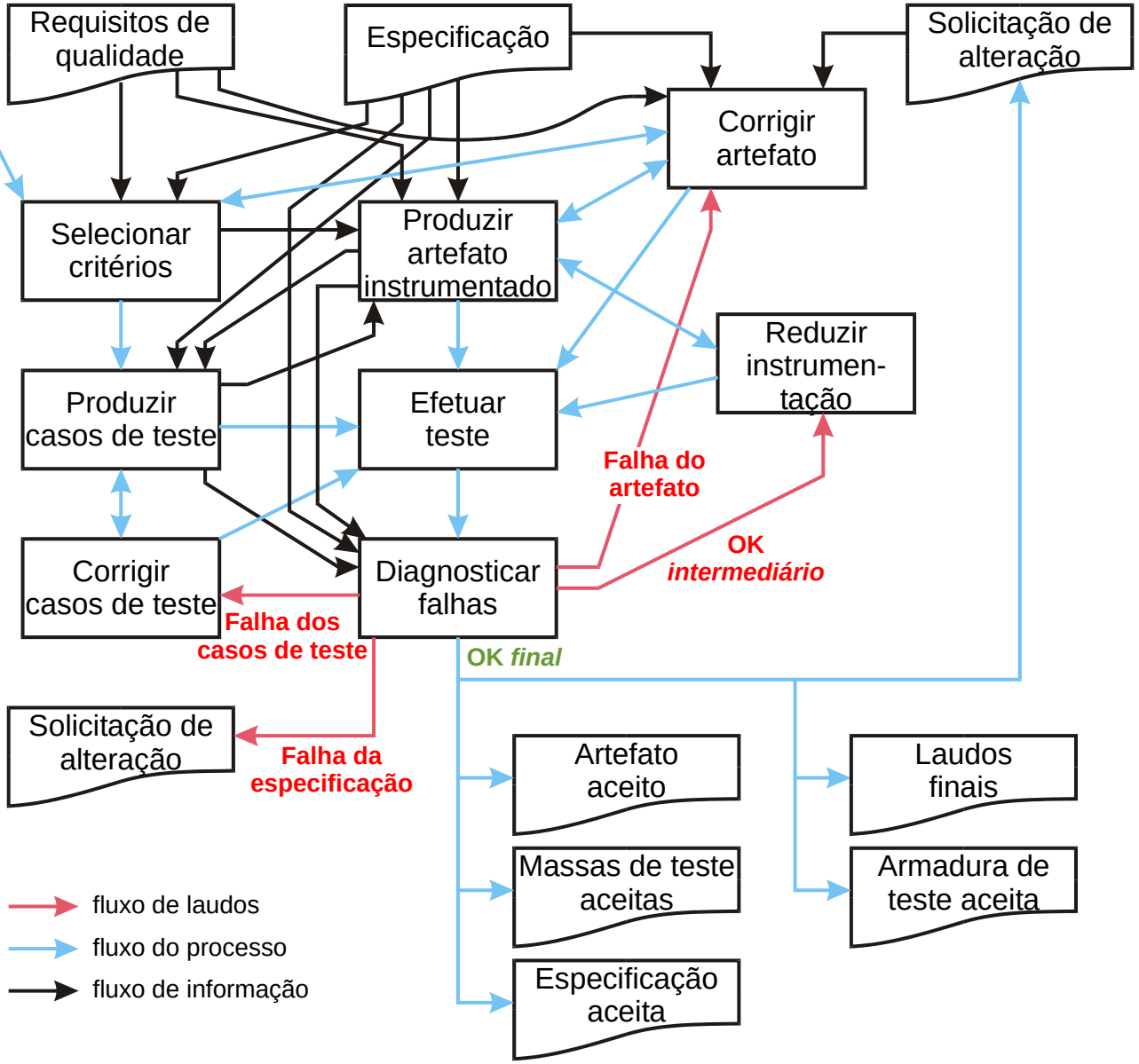
Objetivos do teste: como chegar lá

- Almejar corretude por construção
  - o artefato está sem defeito antes do primeiro teste
  - é um ideal raras vezes alcançado
  - visa minimizar o retrabalho inútil
    - retrabalho inútil tende a ser responsável por até 50% do custo de desenvolvimento
  - requer rigor ao desenvolver

- Almejar corretude por desenvolvimento
  - o artefato está sem defeito antes de ser posto em uso
  - é um ideal que pode ser alcançado
  - visa reduzir o custo total do artefato
    - especialmente: danos e correção após entrega
  - requer rigor ao controlar a qualidade

Processos e Atividades de Teste

Processo de teste



Níveis de abstração dos testes

- Teste de unidade (teste de módulo)
  - examina se um módulo (ou alguns poucos módulos)
    - está em conformidade exata com as suas especificações
      - não faz nem mais nem menos do que o especificado
    - se possui propriedades adequadas ao seu uso
  - ênfase na organização interna dos módulos
- Teste de integração
  - examina se os módulos de um construto se compõem corretamente uns com os outros
  - ênfase nas interfaces entre módulos

Níveis de abstração dos testes

- Teste de programa
  - examina se o programa (conjunto de módulos) satisfaz exatamente as suas especificações
  - ênfase na demonstração que o programa realiza o que foi especificado, do ponto de vista dos desenvolvedores
- Teste de funcionalidade
  - examina se o programa atende as necessidades dos usuários
  - ênfases em
    - verificar adequação do serviço do programa às necessidades e expectativas do usuário
    - verificar a utilizabilidade do programa
    - verificar outros requisitos não funcionais

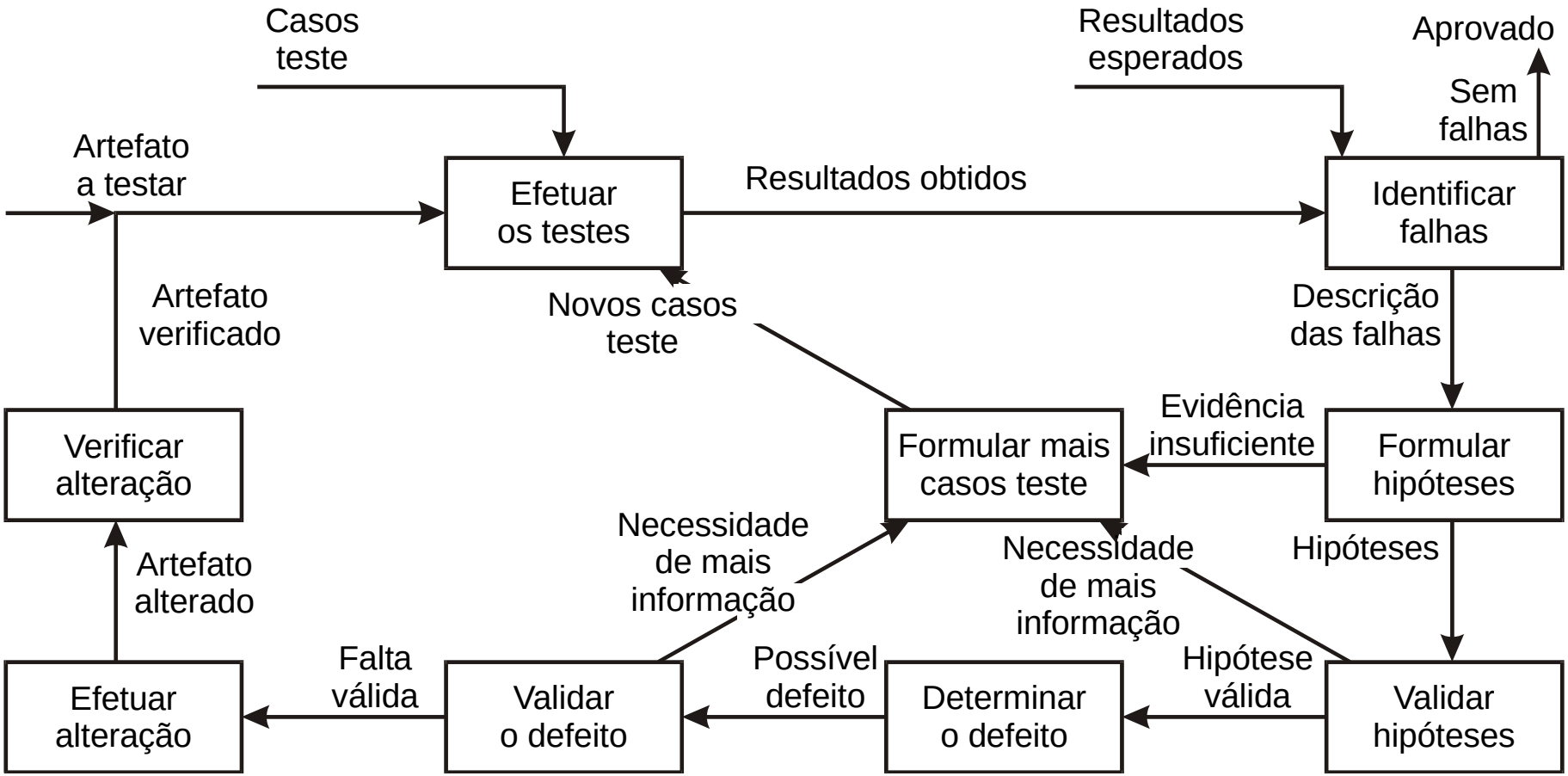
Critérios de seleção de casos de teste

- Critérios de seleção de casos de teste são utilizados para gerar os casos de teste que compõem a massa de teste
  - a geração pode ser manual, ou parcial ou totalmente automatizada
- Categorias de critérios de seleção de casos de teste
  - teste caixa fechada (teste funcional)
    - gera os casos utilizando somente a especificação
    - a massa pode ser desenvolvida antes ou junto com o código
  - teste caixa aberta (teste estrutural)
    - gera os casos de teste utilizando o código completo e a especificação
  - teste de estruturas de dados
    - gera os casos de teste utilizando modelos e/ou o código de declaração da estrutura de dados e a especificação

Diagnose

- A diagnose procura localizar todos os pontos de um ou mais artefatos que constituem o defeito causador de um problema (falha) observado
- Um defeito pode estar
  - concentrado em um único lugar no código
  - espalhado sobre vários lugares de um mesmo artefato sob teste
  - espalhado sobre vários artefatos aprovados ou não
  - espalhado sobre diversos programas, inclusive programas que intuitivamente nada têm a ver com o artefato sob teste
- Um defeito pode ser causado
  - por erro de programação
  - por erro de projeto ou arquitetura
  - por erro de especificação

Diagnose: processo



Depuração

- A depuração (*debugging*) é a atividade de eliminar os defeitos diagnosticados de forma
  - completa
  - correta
  - sem introduzir novos defeitos
    - estatísticas mostram que mais de 50% das vezes são introduzidos novos defeitos
- Evite a experimentação sem objetivo claro
  - somente altere o módulo quando tiver certeza de ter identificado completamente a causa (defeito) da falha observada

Registro de falhas

- O resultado do controle da qualidade é um laudo
- O laudo
  - identifica o artefato sendo controlado
  - registra todas as falhas
    - encontradas durante os testes
    - reportadas durante o uso produtivo
  - registra todos os defeitos
    - encontrados durante a inspeção ou argumentação
  - contém a história de todos os testes realizados

Registro de falhas

Data registro	Id teste	Nível	Sintoma	Data correção	Artefatos alterados	Classe do defeito	Correção realizada

Técnicas e Critérios de Teste

Técnicas de teste

- As técnicas de teste são definidas conforme o tipo de informação utilizada para realizar o teste
  - As diferentes técnicas são complementares!!!!

- Técnica Caixa-Preta
  - Os testes são baseados exclusivamente na especificação do programa.
  - Nenhum conhecimento de como o programa está implementado é requerido.

- Técnica Caixa-Branca
  - Os testes são baseados na estrutura interna do programa, ou seja, na implementação.

Técnicas de teste

- As técnicas de teste são definidas conforme o tipo de informação utilizada para realizar o teste
  - As diferentes técnicas são complementares!!!!

- Técnica Caixa-Preta
  - Os testes são baseados exclusivamente na especificação do programa.
  - Nenhur ograma está implem

- Técnica C
  - Os test programa, ou seja, na implementação.

- Relações entre entradas e saídas
- Assertivas
- Requisitos
- Conhecimento do domínio da aplicação

Como criar os casos de teste?

- Caso de teste é composto de
  - Entrada
  - Saída esperada
  - Saída obtida

- Mas como escolher um conjunto de casos de testes adequados dentre todo o conjunto de entradas válidas e inválidas?
  - É impraticável / impossível testar todas as possíveis entradas, mesmo para funções muito simples
    - Ex.: int soma(int x, int y)

Critérios de teste

- Maneira sistemática e planejada para conduzir os testes
  - Fornece indicações a respeito de quais casos de teste utilizar de modo a aumentar as chances de revelar erros no programa
- Quando erros não forem revelados...
  - Estabelecer um nível elevado de confiança na correção do programa.

Critérios de Teste - Teste Caixa Preta

- Critérios para seleção de entradas:
  - Aleatório
  - Classes de Equivalência
  - Análise de Limite

Testes Caixa Preta – Aleatório

- Cada módulo / sistema possui um conjunto domínio de entradas do qual as entradas de teste são selecionadas
- Se um testador escolhe aleatoriamente (ou ao acaso) entradas de teste deste domínio, isto é chamado de teste aleatório
  - Ex.: soma(2, 3), soma(0, 1)...
- Vantagens:
  - Fácil e de baixo custo
- Desvantagens:
  - Não há garantia da qualidade dos casos de teste

Testes Caixa Preta – Classes de Equivalência

- O critério de Classes de Equivalência divide o domínio das entradas em finitos conjuntos (cada um destes conjuntos é chamado de classe de equivalência)
- O objetivo deste critério é definir um conjunto minimal de casos de teste
  - Para cada classe de equivalência, um caso de teste
    - Se determinado caso de teste não conseguir encontrar uma falha, outros casos de testes equivalentes também não o conseguirão
- O testador deve considerar tanto classes de equivalência válidas como inválidas
- Também é possível definir classes de equivalência para o conjunto de saídas

Testes Caixa Preta – Classes de Equivalência

- Vantagens:
  - Reduz o conjunto de casos de testes
  - Diversifica o conjunto de casos de testes, aumentando a probabilidade de detectar defeitos
- Desvantagens:
  - Não é fácil identificar todas as classes de equivalência

Testes Caixa Preta – Classes de Equivalência

- Exemplo: Uma função que recebe uma string e verifica se esta corresponde a um identificador válido ou não. Um identificador é considerado válido se, e somente se, ele possui no mínimo 3 e no máximo 15 caracteres alfanuméricos, sendo que os dois primeiros caracteres são letras

- Do enunciado podemos derivar três condições básicas (ou assertivas de entrada):
  1. A string só possui caracteres alfanuméricos
  2. O número total de caracteres está entre [3, 15]
  3. Os dois primeiros caracteres são letras

Testes Caixa Preta – Classes de Equivalência

- Da primeira condição derivamos as seguintes classes de equivalência:

  "A string só possui caracteres alfanuméricos

  - Classe 1: A string só possui caracteres alfanuméricos
  - Classe 2: A string possui um ou mais caracteres não alfanuméricos

Testes Caixa Preta – Classes de Equivalência

- Da segunda condição derivamos as seguintes classes de equivalência:

  "O número total de caracteres está entre [3, 15]"

  - Classe 3: A string possui entre [3, 15] caracteres
  - Classe 4: A string possui menos do que 3 caracteres
  - Classe 5: A string possui mais do que 15 caracteres

Testes Caixa Preta – Classes de Equivalência

- Da terceira condição derivamos as seguintes classes de equivalência:

  "Os dois primeiros caracteres são letras"

  - Classe 6: Os dois primeiros caracteres são letras
  - Classe 7: Um dos dois primeiros caracteres não é letra

Testes Caixa Preta – Classes de Equivalência

- Ao fim, temos as seguintes classes de equivalência:

Condição	Classes Válidas	Classes Inválidas
1	Classe 1	Classe 2
2	Classe 3	Classes 4, 5
3	Classe 6	Classe 7

- E o seguinte conjunto de casos de testes:
  - Classe 1: {"ab12345"}
  - Classe 2: {"sah12^*&^"}
  - Classe 3: {"abcdefgh"}
  - Classe 4: {"AA"}
  - Classe 5: {"ABCDE1234567890123456789O"}
  - Classe 6: {"ab12312312"}
  - Classe 7: { "1asd"}

Testes Caixa Preta – Análise de Limites

- Muitos erros comuns de programação ocorrem nas condições limites
  - Ex.: for(i = 0; i **<** x; i++) ou for(i = 0; i **<=** x; i++) ?
- O critério de Análise de Limites tem como objetivo usar entradas próximas aos limites para exercitar a checagem dessas condições
- Geralmente é usada para refinar as entradas criadas com o critério de Classes de Equivalência
- Também pode ser aplicado às saídas

Testes Caixa Preta – Análise de Limites

- Alguns *guidelines*:
  - Se uma condição sobre uma entrada /saída for um intervalo, defina:
    - Para cada limite (superior / inferior), três casos de teste:
      - Um logo abaixo do limite
      - Um logo acima do limite
      - Um em cima do limite
  - Se uma condição sobre uma entrada é do tipo "deve ser" (ou "é", "só pode", etc) defina casos de testes para os casos verdadeiro e falso
  - Se uma condição sobre uma entrada / saída for um conjunto ordenado (lista ou tabela) defina testes que foquem:
    - No primeiro e no último elemento
    - Conjunto vazio

Testes Caixa Preta – Análise de Limites

- Exemplo: Uma função que recebe uma string e verifica se esta corresponde a um identificador válido ou não. Um identificador é considerado válido se, e somente se, ele possui no mínimo 3 e no máximo 15 caracteres alfanuméricos, sendo que os dois primeiros caracteres são letras

- Aplicando o critério de Análise de Limites sobre o tamanho da string (condição 2):
  - Limite inferior: 3
    - Logo abaixo do limite inferior (AbLI): tamanho = 2 (inválido)
    - Logo acima do limite inferior (AcLI): tamanho = 4 (válido)
    - Em cima do (LI): tamanho = 3 (válido?)
  - Limite superior: 15
    - Logo abaixo do limite superior (AbLS): tamanho = 14 (válido)
    - Logo acima do limite superior (AcLS): tamanho = 16 (inválido)
    - Em cima do limite superior (LS): tamanho = 15 (válido?)

Testes Caixa Preta

- Combinando Classes de Equivalência e Análise de Limites:
  - É necessário ter um conjunto de casos de testes que cubra todas as classes de equivalência e todos os limites
  - Mas é importante que o conjunto de casos de testes seja minimal!
    - Se fizer um caso de teste para cada caso, o número aumentará rapidamente
    - É possível criar casos de testes que cubram mais de uma classe de equivalência ao mesmo tempo!

Testes Caixa Preta

- Exemplo: (a tabela está incompleta)

ID	Entrada	Classe válida e limite coberto	Classe inválida e limite coberto
1	“ABC1”	Classe 1, Classe 3(AcLI), Classe 6	
2	“ABC*”	Classe 3(AcLI), Classe 6	Classe 2
3	“A1”	Classe 1	Classe 4(AbLI), Classe 7
…			

FIM