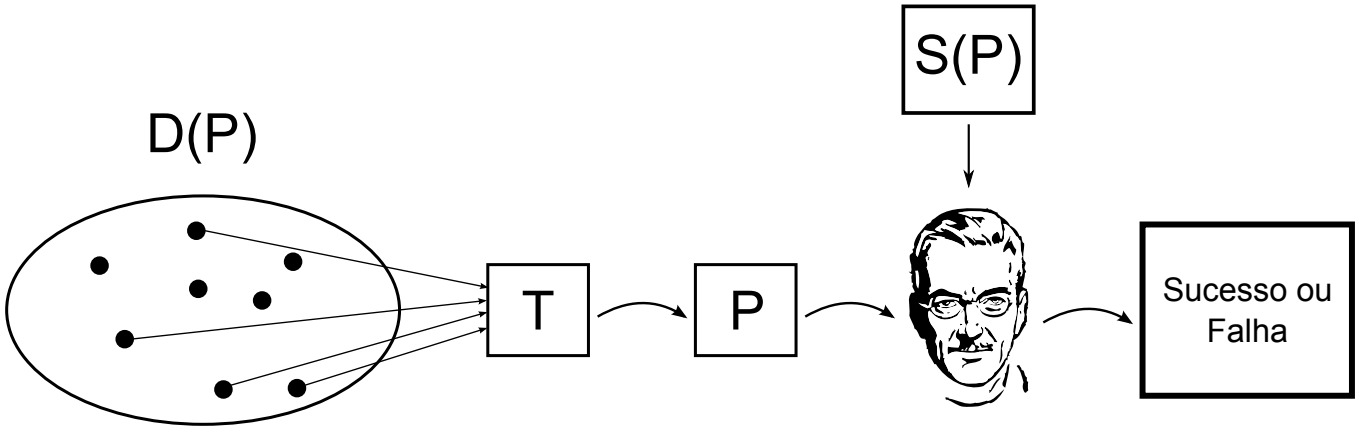
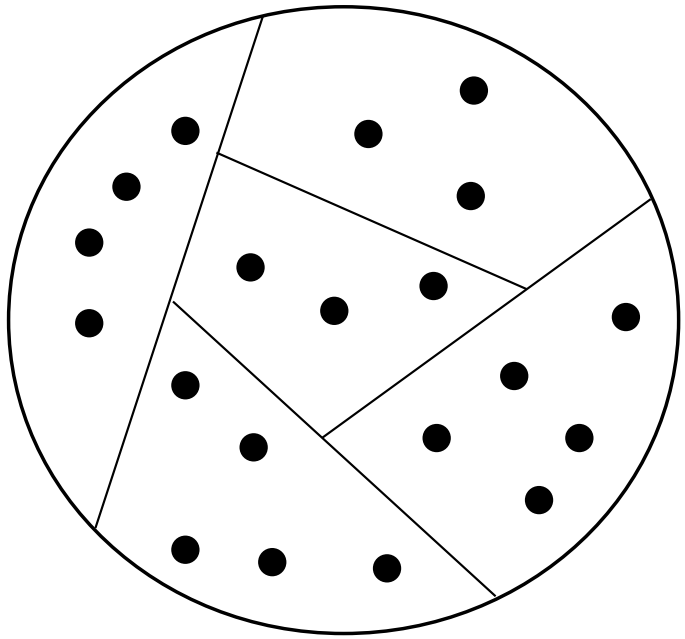
Teste Funcional (caixa-preta)

Prof. Otávio Lemos (UNIFESP)
Prof. Fabiano Ferrari (UFSCar)

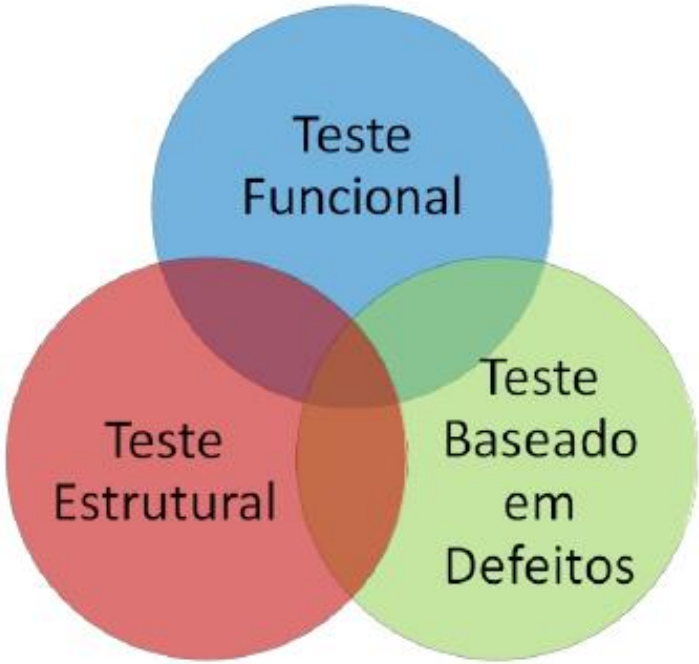
- Relembrar – teste de software, domínio de teste, dado de teste, caso de teste



D(P)

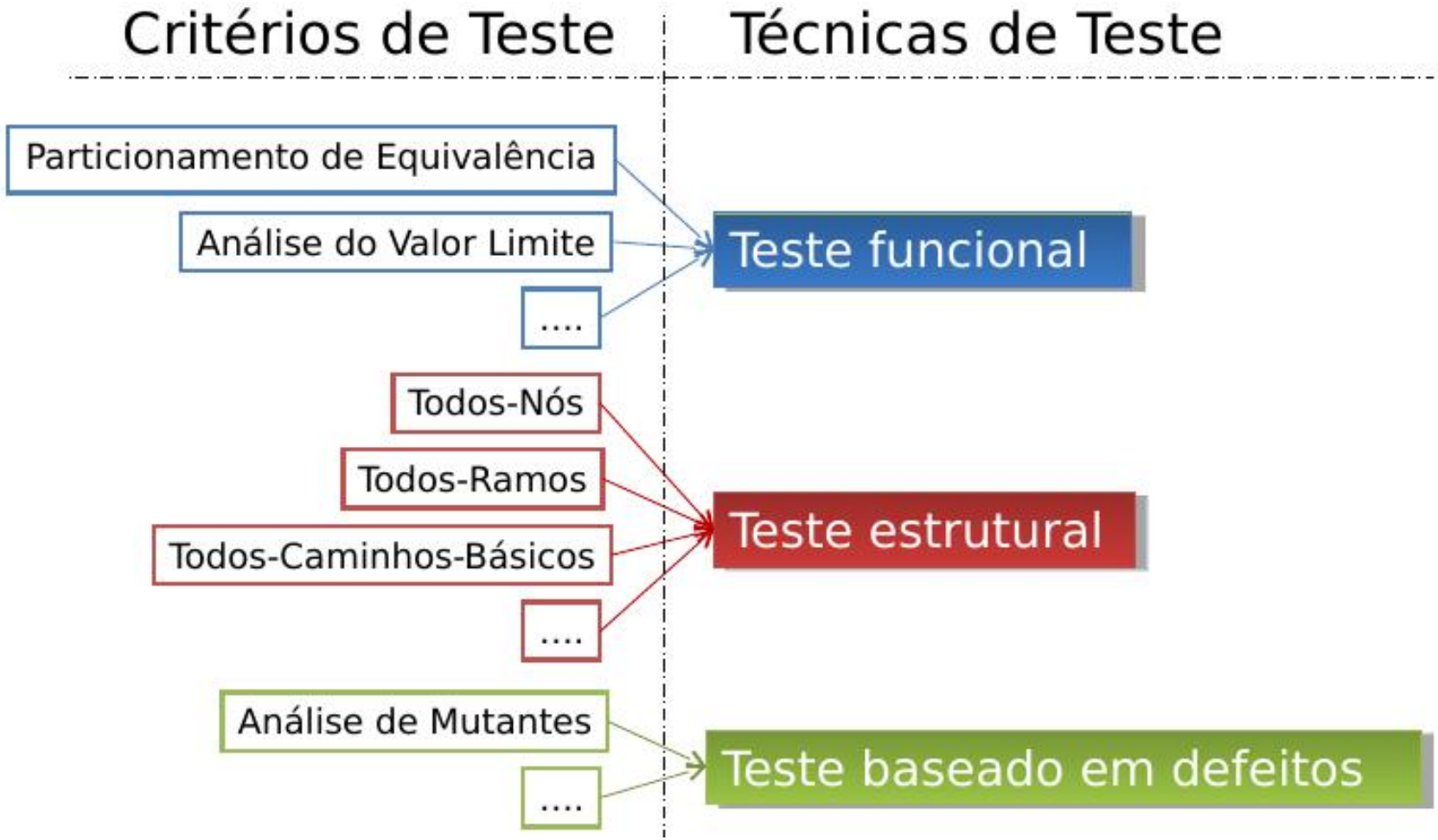


Fonte	Modelo	Técnica
Especificação	Modelo do sistema	Teste funcional
Código-fonte	Modelo do programa	Teste estrutural
Defeitos comuns	Modelo de defeitos	Teste baseado em defeitos

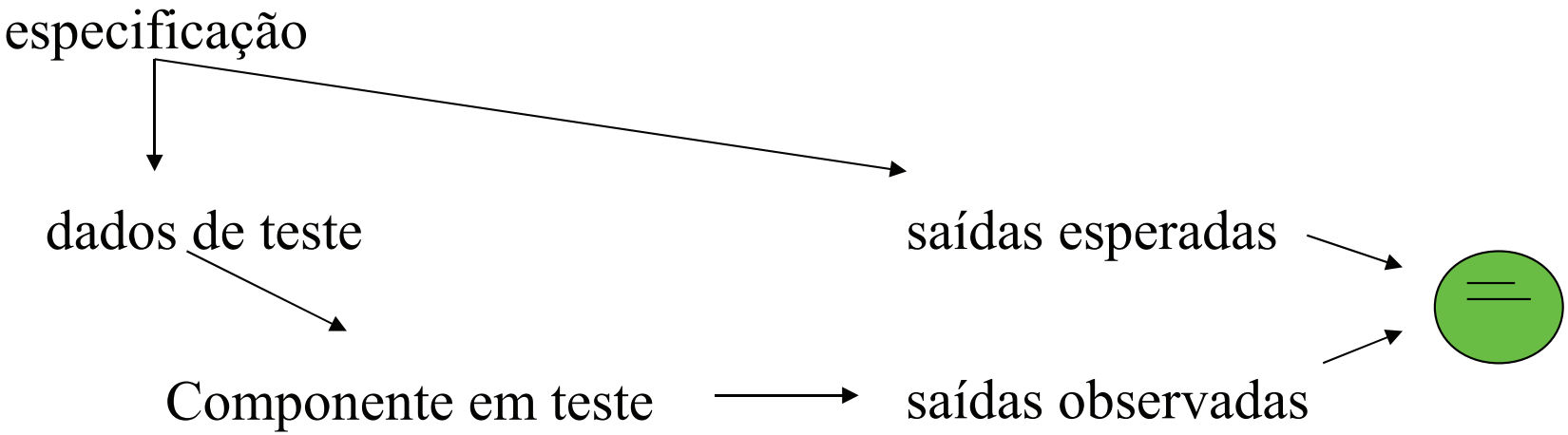


São complementares, pois identificam tipos diferentes de defeitos.

- Seleção e Adequação
  - Seleção: os CTs são criados para satisfazer os requisitos do critério de teste
  - Adequação: os CTs são criados, por exemplo, aleatoriamente, e então se verifica se eles atendem (ou satifazem) os requisitos do critério de teste
- Conjunto de casos de teste C-adequado


Critérios de Teste
Técnicas de Teste
Particionamento de Equivalência
Análise do Valor Limite
....
Teste funcional
Todos-Nós
Todos-Ramos
Todos-Caminhos-Básicos
....
Teste estrutural
Análise de Mutantes
....
Teste baseado em defeitos

- Caixa-preta
- Objetivo do teste: cobertura da especificação
- Modelos de teste obtidos das especificações:
  - Requisitos
  - Arquitetura


especificação
dados de teste
saídas esperadas
Componente em teste
saídas observadas

- Detecção de defeitos depende da especificação
  - especificações ambíguas, incompletas, incorretas e inconsistentes são perigosas para o teste
  - especificações devem ser verificadas (Ex.: inspeção)
- Limitação: não garante cobertura de código (nem mesmo de partes essenciais/críticas)
- Por outro lado: qualquer fase, qualquer paradigma – não leva em conta detalhes de implementação

Critérios de Teste Funcional

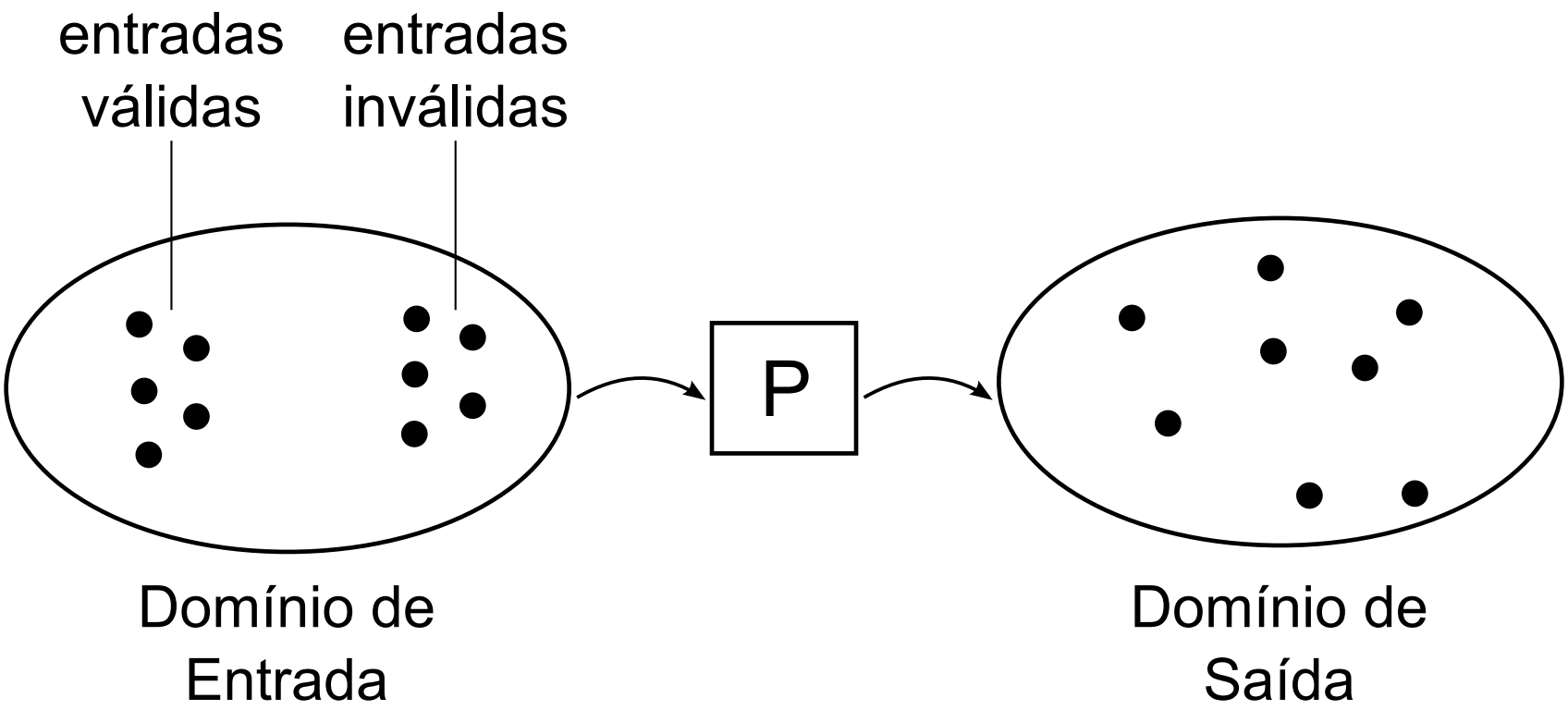
- Particionamento de Equivalência
- Análise de Valor Limite
- Teste Funcional Sistemático
- Grafo Causa-Efeito
- Error-Guessing

Exemplo

Programa "*Cadeia de Caracteres*"

- Descrição:
  O programa solicita do usuário um inteiro positivo no intervalo entre 1 e 20 e então solicita uma cadeia de caracteres desse comprimento. Após isso, o programa solicita um caracter e retorna a posição na cadeia em que o caracter é encontrado pela primeira vez ou uma mensagem indicando que o caracter não está presente na cadeia. O usuário tem a opção de procurar por vários caracteres.

- O domínio de entrada (e algumas vezes de saída) do programa/função é dividido em um número finito de partições (ou classes) de equivalência
  - supõe-se que dados pertencentes a uma partição tenham a capacidade de revelar os mesmos defeitos
  - partições válidas e inválidas são consideradas
- Geração de testes: selecionar um dado de cada partição
- Critério de cobertura: cada partição deve ser considerada por ao menos um CT


entradas
válidas
entradas
inválidas
P
Domínio de
Entrada
Domínio de
Saída

- Ajuda na detecção de classes de equivalência: palavras como ‘intervalo’, ‘conjunto’ – indicação de que dados são processados da mesma forma
- Uma classe representa um conjunto de estados válidos ou inválidos para uma condição de entrada (CE)

- Diretrizes para definição das classes:
  1. Se CE especifica um intervalo de valores – uma classe válida e duas inválidas.
     ⇒ Exemplo: "*a senha pode conter ter no mínimo 8 e no máximo 12 caracteres*"
  2. Se CE especifica um conjunto de valores determinados tratados de maneira diferente – uma válida para cada um; uma inválida com valor qualquer.
     ⇒ Exemplo: "*O veículo pode ser do tipo passeio ou transporte*"
  3. Se CE especifica situação do tipo "deve ser assim" – uma classe válida e uma inválida.
     ⇒ Exemplo: "*A alíquota de imposto deve ser de 8%*"

- Se classes se sobrepõem ou elementos de uma mesma classe devem surtir efeitos diferentes: refinar, reduzir, separar.
- Uma vez identificadas classes de equivalência: criar casos de teste para cobri-las
  - Ideia: cada novo caso de teste deve cobrir o maior número de classes válidas possível
  - Para as inválidas: um caso para cada classe – uma entrada incorreta pode mascarar os efeitos de outra entrada incorreta

- Exemplo de aplicação – Considerando a especificação dada anteriormente (*Cadeia de Caracteres*), têm-se quatro entradas:
  - T - tamanho da cadeia de caracteres
  - CC - uma cadeia de caracteres
  - C - um caracter a ser procurado
  - O - a opção por procurar por mais caracteres

- Pelo domínio de entrada:
  - T: entre 1 e 20 (inclusive);
  - CC e C não determinam classes de equivalência, pois os caracteres podem ser quaisquer
  - O: “sim” ou “não”
- Pelo domínio de saída:
  - a posição na qual o caracter foi encontrado na cadeia de caracteres
  - mensagem “caracter não foi encontrado”

Variável de entrada	Classes válidas	Classes inválidas
T	1 <= T <= 20 (1)	T < 1 (2); T > 20 (3)
O	Sim (4); Não (5)	
Variável de saída		
Mensagem	Caracter aparece na posição X da cadeia (6); Caracter não pertence à cadeia (7)	

- Identificadas as classes de equivalência, escolhem-se, arbitrariamente, elementos de cada classe e os casos de teste podem ser definidos segundo a tabela do próximo slide

Variáveis de entrada				Saída esperada
T	CC	C	O	
34				<aguarda nova entrada>
0				<aguarda nova entrada>
3	abc	c	s	Caracter aparece na posição 3 da cadeia
3	abc	k	n	Caracter não pertence à cadeia

- Avaliação do critério:
  - Força – redução no tamanho do domínio de entrada e na criação de dados de teste baseados na especificação
  - Adequado para aplicações com variáveis de entrada facilmente identificadas e com valores específicos
  - Entretanto: critério não é facilmente aplicável quando o domínio de entrada é simples mas o processamento é complexo
  - Problema: embora a especificação sugira que um grupo de dados seja processado de forma idêntica, na prática isso pode não acontecer
  - Além disso: há existem diretrizes "universais" para a determinação dos dados de teste e para encontrar combinações entre eles

Exercício 1

- Defina classes de equivalência e um conjunto de casos de teste adequado para o critério particionamento por classes de equivalência para o programa "*Identificador Silly Pascal*", com a seguinte descrição:

  O programa deve determinar se um identificador é válido ou não em Silly Pascal (uma variante do Pascal). 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.

Exercício 2

- Implemente em Java o programa *identificador Silly Pascal*. A implementação pode ser realizada como um método estático.

Exercício 3

- Implemente em Java o programa *Cadeia de Caracteres*. A implementação pode ser realizada como um método estático.