Engenharia de Software II / Qualidade e Teste de Software

Aula 07: Testes Funcionais (Caixa-Preta)

Breno Lisi Romano

http://sites.google.com/site/blromano

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Bacharelado em Ciência da Computação – BCC (ENSC6)

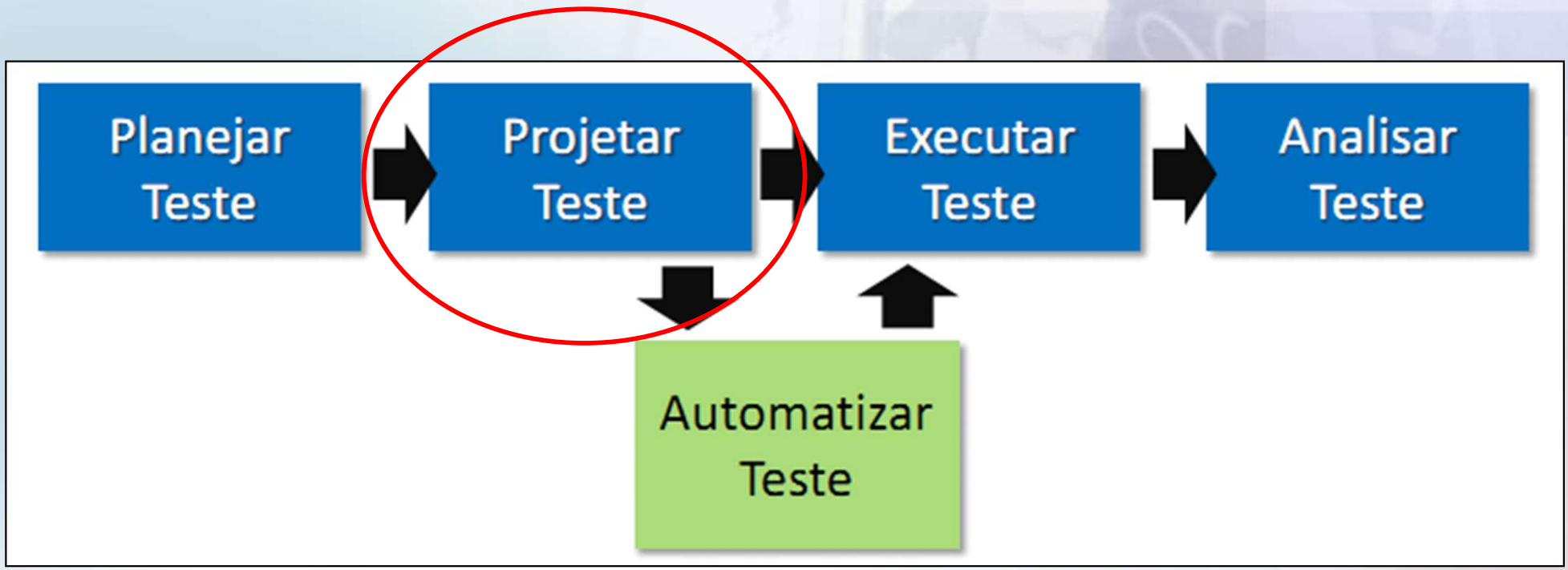
Tecnologia em Sistemas para Internet – TSI (QTSI6)

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Revisão: Teste de Software

- Representam uma oportunidade de detectar **defeitos** antes do software ser entregue aos usuários
- A atividade de testes pode ser feita de forma **manual** e/ou **automática** e tem por objetivos:
  - **Detectar Erros** para **Eliminar** os **Defeitos** e **Evitar** as **Falhas**
  - **Produzir casos de teste** que tenham elevadas probabilidades de **revelar um defeito ainda não descoberto**, com uma quantidade mínima de tempo e esforço
  - **Comparar o resultado dos testes com os resultados esperados** → produzir uma indicação da qualidade e da confiabilidade do software. Quando há diferenças, inicia-se um processo de depuração para descobrir a causa

Revisão: Processo Básico de Teste de Software



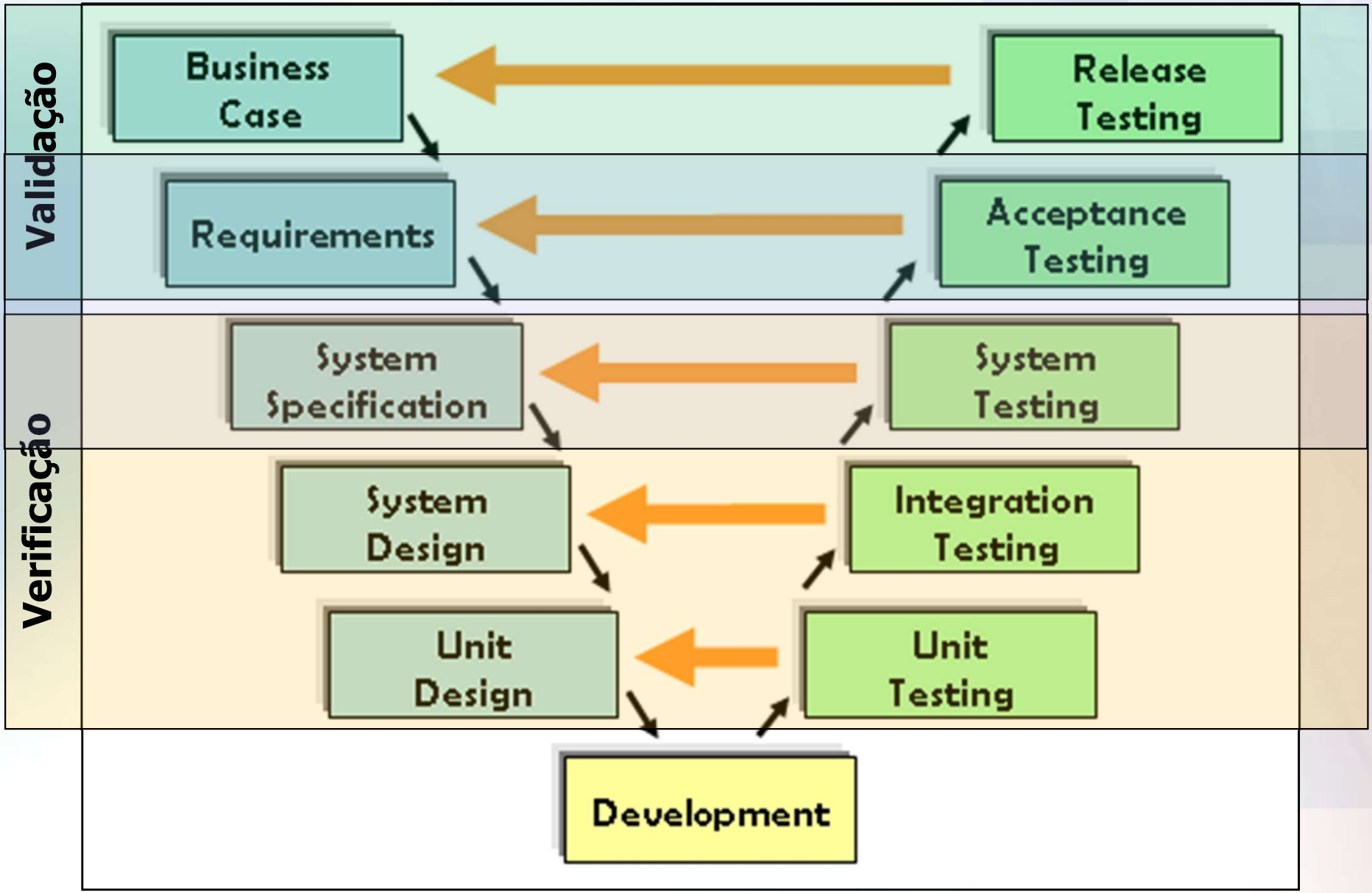
Revisão: V&V

- **Verificação:**
  - Processo de avaliação de um sistema ou componente para determinar se os **artefatos** produzidos **satisfazem** às **especificações** determinadas no início da fase
  - **"Você construiu corretamente?"**

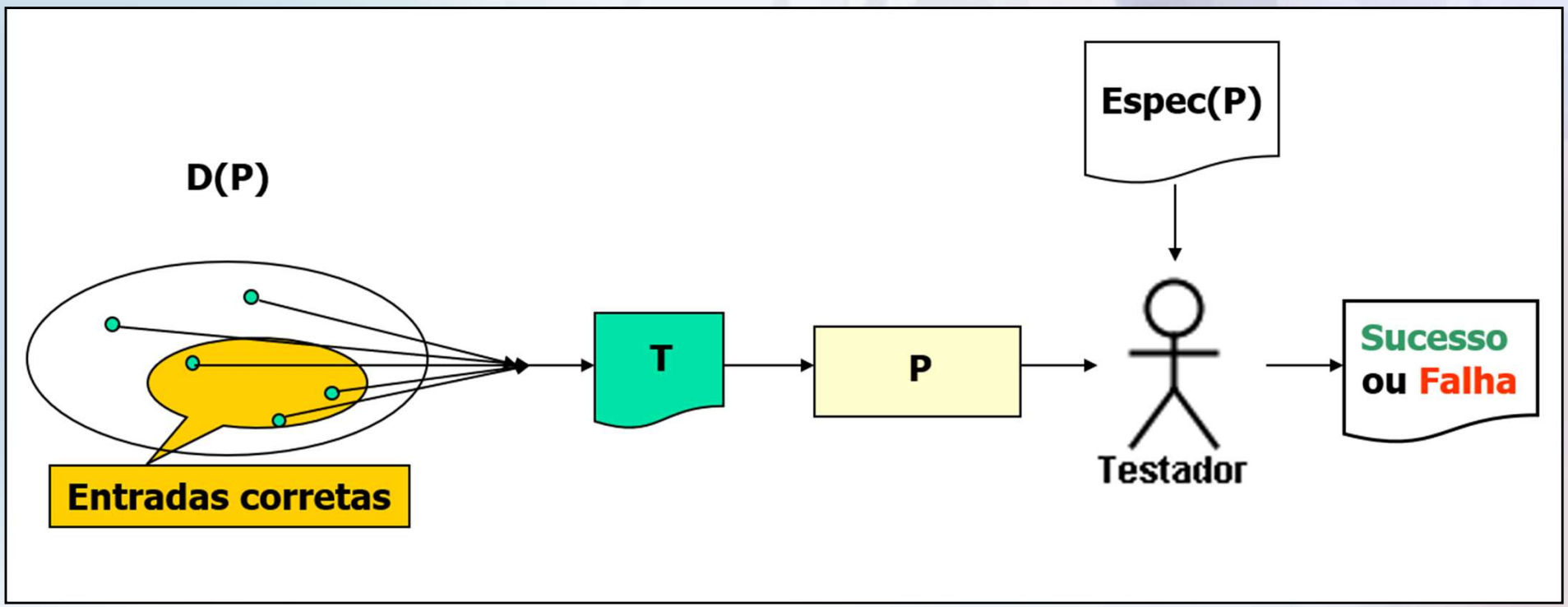
- **Validação:**
  - Processo de avaliação para determinar se o **sistema atende** as **necessidades** e **requisitos** dos usuários
  - **"Você construiu o sistema correto?"**

- **Testes:**
  - Processo de exercitar um sistema ou componente para **validar** que este **satisfaz** os **requisitos** e para **verificar para identificar defeitos**

Revisão: Modelo V



Revisão: Cenário Típico da Atividades de Teste



Revisão: Teste de Software

- Existem duas maneiras de se selecionar elementos de cada um dos **subdomínios de teste** (Delamaro et al., 2007):
  - **Teste Aleatório:** um grande número de casos de teste é selecionado aleatoriamente, de modo que, probabilisticamente, se tenha uma boa chance de ter todos os subdomínios representados em **T**
  - **Teste de Subdomínios ou Teste de Partição:** procura-se estabelecer quais são os subdomínios a serem utilizados e, então, selecionam-se os casos de teste em cada subdomínio

Revisão: Teste de Subdomínios

- A identificação dos subdomínios é feita com base em **critérios de teste**
- Dependendo dos critérios estabelecidos, são obtidos subdomínios diferentes.
- Tipos principais de **critérios de teste**:
  - Funcionais *
  - Estruturais *
  - Baseados em Modelos
  - Baseados em Defeitos *
- O que diferencia cada um deles é o tipo de informação utilizada para estabelecer os subdomínios (Delamaro et al., 2007) → Técnicas de Teste.

Teste Funcional (1)

- Técnica de **Projeto de Casos de Teste** na qual o sistema é considerado uma caixa-preta
  - Para testá-lo, são **fornecidas entradas e avaliadas as saídas geradas→ verificar** se estão em **conformidade** com os **objetivos** especificados

- Estabelece **critérios para particionar o domínio de entrada em subdomínios**, a partir dos quais serão definidos os casos de teste (Delamaro et al., 2007)

Teste Funcional (2)

- Alguns critérios:
  - **Particionamento de Equivalência ***
  - **Análise de Valor Limite ***
  - **Teste Funcional Sistemático ***
  - Tabela de Decisão
  - *Pairwise*
  - Análise de Domínio

- Todos os critérios das técnicas funcionais baseiam-se apenas na **especificação do produto testado** e, portanto, o Teste Funcional depende fortemente da **Qualidade da Especificação de Requisitos**

- IMPORTANTE: Podem ser aplicados em todas as fases de teste e são independentes de paradigma. Por quê?
  - Não levam em consideração a implementação

Particionamento de Equivalência

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Particionamento de Equivalência (1)

- Divide o **Domínio de Entrada em Classes de Equivalência**, de acordo com a Especificação do Sistema - SUT (*Software Under Test*)

- Uma **Classe de Equivalência** representa: "**um conjunto de estados válidos ou inválidos para cada uma das condições de entrada**"
  - Caso as Classes de Equivalência se sobreponham ou os elementos de uma mesma classe não se comportem da mesma maneira, elas devem ser revistas, a fim de torná-las distintas, ou seja, definir um novo conjunto de classes de equivalência (Delamaro et al., 2007)

Particionamento de Equivalência (2)

- Uma vez **identificadas as classes de equivalência**, devem-se **definir os casos de teste**, **escolhendo um elemento de cada classe**

- **Qualquer elemento da classe pode ser considerado um representante** desta e, portanto, basta ser definido **um caso de teste por classe de equivalência** (Delamaro et al., 2007)

Particionamento de Equivalência (3)

- Passos:
  1. **Identificar as Classes de Equivalência**, tomando por base:
     1. Especificação
     2. As entradas possíveis
     3. As saídas possíveis
     4. Identificar as Classes de Equivalência Válidas e Inválidas
  2. **Gerar Casos de Teste selecionando um elemento de cada classe** e **uma saída possível**, de modo a ter o **menor número possível de casos de teste** (Delamaro et al., 2007)
  3. Casos de Testes adicionais podem ser criados, para que exista um sentimento de conforto com os resultados dos testes
  4. Dificilmente defeitos adicionais serão descobertos com estes casos de teste

Exemplo 01: Cadeia de Caracteres (1)

- **Especificação:**
  - **Curso Normal:** O usuário informa uma cadeia de caracteres contendo de 1 a 20 caracteres e um caractere a ser procurado. O sistema informa a posição na cadeia em que o caractere foi encontrado pela primeira vez.
  - **Curso Alternativo:** O caractere informado não está presente na cadeia previamente informada.
    - Uma mensagem é exibida informando que o caractere não pertence à cadeia informada.
  - **Curso de Exceção:** A cadeia informada está vazia ou seu tamanho é maior do que 20.
    - Uma mensagem de erro é exibida indicando que a cadeia de caracteres deve ter tamanho de 1 a 20.
  - **Curso de Exceção:** Não é informado um caractere a ser procurado ou é informado mais de um caractere.
    - Uma mensagem de erro é exibida indicando que um caractere a ser procurado deve ser informado e ser apenas um caractere.

Exemplo 01: Cadeia de Caracteres (2)

- **Entradas:**
  - Cadeia de caracteres
  - Caractere a ser procurado

- **Saídas possíveis:**
  - A posição da primeira ocorrência do caractere a ser procurado na cadeia
  - Mensagem informando que o caractere não pertence à cadeia informada
  - Mensagem de erro informando que a cadeia é inválida (Tamanho da Cadeia incorreto!)
  - Mensagem de erro informando que o caractere a ser procurado é inválido (Vazio ou Mais que um caractere)

Exemplo 01: Cadeia de Caracteres (3)

- **Classe de Equivalência:**

Entrada	Classes de Equivalência Válidas	Classes de Equivalência Inválidas
Cadeia de caracteres	Qualquer cadeia de tamanho T, tal que $1 \leq T \leq 20$	Qualquer cadeia de tamanho T, tal que $T < 1$ ou $T > 20$
Caractere a ser procurado	Caractere de t=1 e pertence à cadeia	Caractere não informado
	Caractere de t=1 não pertence à cadeia	Mais de um caractere é informado (t>1)

Exemplo 01: Cadeia de Caracteres (4)

- **Casos de Teste:**

Caso de Teste	Cadeia de Caracteres	Caractere a ser procurado	Saída Esperada
CT #01	abc	b	2
CT #02	abc	d	O caractere não pertence à cadeia informada.
CT #03	Abcdefghijklm nopqrstuvxyz	<<qualquer>>	Erro: Cadeia inválida.
CT #04	Abc	xyz	Erro: Mais de um caractere informado (Caractere inválido).
CT #05	abc	<<caractere não informado>>	Erro: Nenhum caractere informado (Caractere inválido).

Exemplo 02: Fibonacci (1)

- **Especificação:**
  - **Curso Normal:** O usuário informa um número n, inteiro maior do que zero. O enésimo termo da série de Fibonacci é calculado e apresentado. A série de Fibonacci tem a seguinte formação:
    - O dois primeiros termos da série são iguais a 1 (F0 e F1).
    - O enésimo termo (n > 1) é calculado da seguinte forma: Fn = Fn-1 + Fn-2.
    - Portanto, a série de Fibonacci tem a seguinte forma: 1,1,2,3,5,8,13,21,34...
  - **Curso de Exceção:** O número informado é menor do que 0.
    - Uma mensagem de erro é exibida indicando que o número deve ser maior ou igual do que 0
  - **Curso de Exceção:** O valor informado não é um número inteiro.
    - Uma mensagem de erro é exibida indicando que um número inteiro positivo deve ser informado

Exemplo 02: Fibonacci (2)

- **Entradas:**
  - Número inteiro n
- **Saídas possíveis:**
  - Enésimo termo da série de Fibonacci
  - Erro: n deve ser maior/igual do que 0
  - Erro: o valor informado deve ser um número inteiro positivo

Exemplo 03: Fibonacci (3)

- **Classes de Equivalência:**

Entrada	Classes de Equivalência Válidas	Classes de Equivalência Inválidas
n	n é um número inteiro positivo.	n é um número inteiro menor que 0.
		n não é um número inteiro.

Exemplo 03: Fibonacci (4)

- **Casos de Teste:**

Caso de Teste	n	Saída Esperada
CT #01	6	13
CT #02	-2	Erro: n deve ser maior/igual do que 0.
CT #03	a	Erro: Entre com um número inteiro positivo.

Avaliação do Critério

- Possibilita uma **boa redução do tamanho do domínio** de **entrada**
- É **especialmente adequado** para aplicações em que as **variáveis de entrada podem ser facilmente identificadas** e assumem valores específicos
- Não é tão facilmente aplicável quando o domínio de entrada é simples, mas o processamento é complexo
  - Nestes casos, a especificação pode sugerir que um grupo de dados seja processado de forma idêntica, mas isso pode não ocorrer na prática

Análise do Valor Limite

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Análise do Valor Limite (1)

- Critério usado **em conjunto com o particionamento de equivalência**
- **Premissa: Casos de Teste que exploram condições limites** têm **maior probabilidade de encontrar defeitos**
- Tais **condições** estão **exatamente sobre ou imediatamente acima ou abaixo dos limitantes** das Classes de Equivalência (Delamaro et al., 2007)

Análise do Valor Limite (2)

- **Recomendações:**
  - Condição de entrada especificando um intervalo de valores:
    - Definir casos de teste para os limites desse intervalo (inferior e superior) e para valores imediatamente subsequentes, acima e abaixo, que explorem as classes vizinhas
  - Condição de entrada especificando uma quantidade de valores:
    - Definir casos de teste com nenhum valor de entrada, somente um valor, com a quantidade máxima de valores e com a quantidade máxima de valores + 1
  - Aplicar as recomendações acima para condições de saída, quando couber
  - Se a entrada ou a saída for um conjunto ordenado, deve ser dada atenção especial aos primeiro e último elementos desse conjunto

Exemplo 01: Cadeia de Caracteres (1)

- **Especificação:**
  - **Curso Normal:** O usuário informa uma cadeia de caracteres contendo de 1 a 20 caracteres e um caractere a ser procurado. O sistema informa a posição na cadeia em que o caractere foi encontrado pela primeira vez.
  - **Curso Alternativo:** O caractere informado não está presente na cadeia previamente informada.
    - Uma mensagem é exibida informando que o caractere não pertence à cadeia informada.
  - **Curso de Exceção:** A cadeia informada está vazia ou seu tamanho é maior do que 20.
    - Uma mensagem de erro é exibida indicando que a cadeia de caracteres deve ter tamanho de 1 a 20.
  - **Curso de Exceção:** Não é informado um caractere a ser procurado ou é informado mais de um caractere.
    - Uma mensagem de erro é exibida indicando que um caractere a ser procurado deve ser informado e ser apenas um caractere.

Exemplo 01: Cadeia de Caracteres (2)

- **Entradas:**
  - Cadeia de caracteres
  - Caractere a ser procurado

- **Saídas possíveis:**
  - A posição da primeira ocorrência do caractere a ser procurado na cadeia
  - Mensagem informando que o caractere não pertence à cadeia informada
  - Mensagem de erro informando que a cadeia é inválida (Tamanho da Cadeia incorreto!)
  - Mensagem de erro informando que o caractere a ser procurado é inválido (Vazio ou Mais que um caractere)

Exemplo 01: Cadeia de Caracteres (3)

- **Classe de Equivalência:**

Entrada	Classes de Equivalência Válidas	Classes de Equivalência Inválidas
Cadeia de caracteres	Qualquer cadeia de tamanho T, tal que $1 \leq T \leq 20$	Qualquer cadeia de tamanho T, tal que $T < 1$ ou $T > 20$
Caractere a ser procurado	Caractere de t=1 e pertence à cadeia	Caractere não informado
	Caractere de t=1 não pertence à cadeia	Mais de um caractere é informado (t>1)

Exemplo 01: Cadeia de Caracteres (4)

- **Casos de Teste Adicionais – Análise do Valor Limite:**

Caso de Teste	Cadeia de Caracteres	Caractere a ser procurado	Saída Esperada
CT #06	[VAZIA] = NULL	<<qualquer>>	Erro: Cadeia inválida. Entre com uma cadeia contendo de 1 a 20 caracteres.
CT #07	abcdefghijklm nopqrstu	<<qualquer>>	Erro: Cadeia inválida. Entre com uma cadeia contendo de 1 a 20 caracteres.
CT #08	a	a	1
CT #09	a	x	O caractere não pertence à cadeia informada.

Exemplo 01: Cadeia de Caracteres (5)

- **Casos de Teste Adicionais – Análise do Valor Limite:**

Caso de Teste	Cadeia de Caracteres	Caractere a ser procurado	Saída Esperada
CT #10	abcdefghijklm nopqrs	a	1
CT #11	abcdefghijklm nopqrst	t	20
CT #12	ab	xy	Erro: Mais de um caractere informado.
CT #13	abc	[VAZIA] = NULL	Erro:Caractere a ser procurado é inválido.

Exemplo 02: Fibonacci (1)

- **Especificação:**
  - **Curso Normal:** O usuário informa um número n, inteiro maior do que zero. O enésimo termo da série de Fibonacci é calculado e apresentado. A série de Fibonacci tem a seguinte formação:
    - O dois primeiros termos da série são iguais a 1 (F0 e F1).
    - O enésimo termo (n > 1) é calculado da seguinte forma: Fn = Fn-1 + Fn-2.
    - Portanto, a série de Fibonacci tem a seguinte forma: 1,1,2,3,5,8,13,21,34...
  - **Curso de Exceção:** O número informado é menor do que 0.
    - Uma mensagem de erro é exibida indicando que o número deve ser maior ou igual do que 0
  - **Curso de Exceção:** O valor informado não é um número inteiro.
    - Uma mensagem de erro é exibida indicando que um número inteiro positivo deve ser informado

Exemplo 02: Fibonacci (2)

- **Entradas:**
  - Número inteiro n
- **Saídas possíveis:**
  - Enésimo termo da série de Fibonacci
  - Erro: n deve ser maior/igual do que 0
  - Erro: o valor informado deve ser um número inteiro positivo

Exemplo 03: Fibonacci (3)

- **Classes de Equivalência:**

Entrada	Classes de Equivalência Válidas	Classes de Equivalência Inválidas
n	n é um número inteiro positivo.	n é um número inteiro menor que 0.
		n não é um número inteiro.

Exemplo 03: Fibonacci (4)

- **Casos de Teste Adicionais – Análise do Valor Limite:**

Caso de Teste	n	Saída Esperada
CT #04	0	1
CT #05	1	1
CT #06	-1	Erro: n deve ser maior/igual do que 0.

Como não existe limite superior, tratamos apenas o limite inferior!

Avaliação do Critério

- As vantagens e desvantagens deste critério são as mesmas do critério Particionamento de Equivalência:
  - Possibilita uma boa redução do tamanho do domínio de entrada
  - É especialmente adequado para aplicações em que as variáveis de entrada podem ser facilmente identificadas e assumem valores específicos
  - Não é tão facilmente aplicável quando o domínio de entrada é simples, mas o processamento é complexo

Teste Funcional Sistemático

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Teste Funcional Sistemático (1)

- Combina os critérios de Particionamento de Equivalência e Análise de Valor Limite

- Uma vez que os domínios de entrada e de saída tenham sido particionados, **este critério requer ao menos dois casos de teste de cada partição para minimizar o problema de defeitos coincidentes que mascaram falhas** (Delamaro et al., 2007).

Teste Funcional Sistemático (2)

- Algumas **diretrizes adicionais**:
  - Valores numéricos – Domínio de Entrada:
    - Discretos: testar todos os valores
    - Intervalo de valores: testar os extremos e um valor no interior do intervalo
  - Valores numéricos – Domínio de Saída:
    - Discretos: gerar cada um dos valores
    - Intervalo de valores: gerar cada um dos extremos e ao menos um valor no interior do intervalo
  - Valores ilegais – também devem ser incluídos nos casos de teste para se verificar se o software os rejeita
    - Diretrizes do critério de análise de valor limite devem ser empregadas
  - Tipos de valores diferentes:
    - Explorar tipos ilegais que podem ser interpretados como valores válidos (p.ex., valor real para um campo inteiro)
    - Explorar entradas válidas, mas que podem ser interpretadas como tipos ilegais (p.ex., números em campos que requerem caracteres)

Teste Funcional Sistemático (3)

- **Resumindo:**

1. Testar ao menos dois elementos de uma mesma Classe de Equivalência
2. Testar tipos ilegais de valores para o problema em questão (Ex.: integer, float, double, long, ...)
3. Testar entradas válidas que podem ser consideradas ilegais (Caracteres Especiais)
4. Se preocupar com as saídas dos testes

Exemplo 01: Cadeia de Caracteres

- **Casos de Teste Adicionais – Sistemático:**

Cadeia de Caracteres	Caractere a ser procurado	Saída Esperada
!	` ´	O caractere não pertence à cadeia informada.
!"#$%&()*+´ /01234567	!	1
	&	6
	´	11
	6	19
	7	20

Exemplo 02: Fibonacci

- **Casos de Teste Adicionais – Sistemático:**

n	Saída Esperada
1.0	Erro: Entre com um número inteiro positivo.
-1	Erro: n deve ser maior do que 0.
xyz	Erro: Entre com um número inteiro positivo.
23	28657
24	46368
11	89

Avaliação do Critério

- Por ser baseado nos critérios Particionamento de Equivalência e Análise de Valor Limite, apresenta os mesmos problemas que estes

- **Para tratar o problema da correção coincidente, enfatiza a seleção de mais de um caso de teste por partição ou limite, aumentando, assim, a chance de revelar defeitos** (Delamaro et al., 2007)

Testes Funcionais: Considerações Finais

- Como os **Critérios Funcionais** se baseiam apenas na **Especificação de Requisitos**, **não** podem **assegurar** que **partes críticas e essenciais do código** tenham sido **cobertas**
- Além disso, os próprios **critérios apresentam limitações**
- Assim, é **fundamental que outras técnicas sejam aplicadas em conjunto**, para que o software seja explorado por diferentes pontos de vista (Delamaro et al., 2007)
- Por outro lado, **são as técnicas mais utilizadas no Teste de Software por propiciarem uma maior eficiência nos testes → Encontrar defeitos com baixo custo**

Engenharia de Software II / Qualidade e Teste de Software

Aula 07: Testes Funcionais (Caixa-Preta)

Breno Lisi Romano

Dúvidas?

http://sites.google.com/site/blromano

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Bacharelado em Ciência da Computação – BCC (ENSC6)

Tecnologia em Sistemas para Internet – TSI (QTSI6)

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista