



PARTE 1 - TEORIA BÁSICA

UNIDADE 2: TESTES FUNCIONAIS

Professor: Fabrício Galende Marques de Carvalho
Curso: Banco de Dados
Turno: Noite
Carga horária: 80h

São José dos Campos - 2019

1



OBJETIVOS DA UNIDADE

- Definir testes funcionais.
- Discutir diferentes critérios de testes funcionais.

2

CONTEÚDO DA UNIDADE

1. TESTES FUNCIONAIS
 2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES
 3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES
 4. TESTE FUNCIONAL SISTEMÁTICO
 5. GRAFOS CAUSA-EFEITO
 6. ERROR GUESSING
 7. TABELA DE DECISÃO
 8. MATRIZ DE RASTREABILIDADE
-

3

1. TESTES FUNCIONAIS

1.1. CARACTERÍSTICAS

- Desconsidera detalhes de implementação (“black box”);
 - Considera o ponto de vista do usuário;
 - Considera as especificações do programa para determinação dos dados de teste (e.g.: requisitos funcionais).
-

4

1. TESTES FUNCIONAIS

1.2. VANTAGENS ✓ E DESVANTAGENS ✗

- ✓ Pode ser aplicado em qualquer fase do ciclo de vida de desenvolvimento;
 - ✓ Independe de tecnologia (e.g.: linguagem de programação);
 - ✗ Pode “ignorar” partes críticas do código-fonte;
 - ✗ Depende fortemente de especificação.
-

5

2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.1. PARTIÇÃO DE EQUIVALÊNCIA

- Nesse critério, uma **partição** corresponde à definição de **subdomínios do programa** que, em teoria, podem ser tratados da mesma maneira por produzirem **efeitos “similares”**.
 - Dessa forma, dados na mesma partição são considerados equivalentes sob o ponto de vista de teste (daí vem o nome partição de equivalência). Pertencem à mesma classe.
-

6

2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.2. OBJETIVO DA TÉCNICA

- ✓ Definir as partições equivalentes para um dado sistema.
- ✓ Selecionar ao menos um dado de teste pertencente a cada partição.

7

2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.2. OBJETIVO DA TÉCNICA (continuação)



8

2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.3. COMO DETERMINAR CLASSES (CATEGORIAS) EQUIVALENTES?

a) **Entrada do sistema é um intervalo:** Uma classe válida e duas inválidas;

b) **Entrada do sistema é uma quantidade de valores:** Uma classe válida e duas inválidas;

9

2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.3. COMO DETERMINAR CLASSES (CATEGORIAS) EQUIVALENTES?

c) **Entrada do sistema é condição do tipo “*deve ser de tal forma*”:** Uma classe válida e uma inválida.

d) **Entrada do sistema com diversos valores, cada um manipulado de uma maneira diferente:** uma classe válida para cada valor e uma inválida para outro valor qualquer.

10

2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.4. EXEMPLO

Programa que aceita como entrada uma cadeia de caracteres contendo entre 1 e 5 caracteres (inclusive) e determina a primeira ocorrência do caractere @.

CLASSES DE ENTRADAS INVÁLIDAS:

Classe inválida 1: Tamanho = 0

Classe inválida 2: Tamanho > 5

11

2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.4. EXEMPLO (continuação)

CLASSES DE ENTRADAS VÁLIDAS



Classe válida: $1 \leq \text{Tamanho} \leq 5$

Subclasse válida 1: Contém um @

Subclasse válida 2: Não contém @

Subclasse válida 3: Contém mais de 1 @

12



2. CRITÉRIO DE PARTIÇÕES EQUIVALENTES

2.4. EXEMPLO (continuação)

Esquemáticamente:

Categoria válida 1: Exemplo CT: <"xy@" , "posição 2" >	Categoria inválida 1: Exemplo CT: <"", "entrada inválida">
Categoria válida 2: Exemplo CT: <"x@@" , "posição 1">	
Categoria válida 3: Exemplo CT: <"xyz" , "não achado">	
Categoria inválida 2: Exemplo CT: <"abcdefg" , "entrada inválida">	

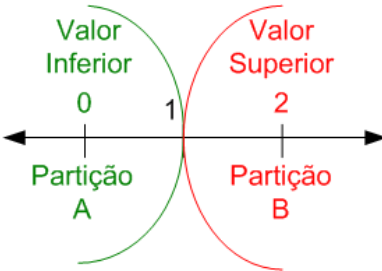
13

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.1. INTRODUÇÃO

Considerando uma determinada partição da entrada, os valores que estão exatamente em cima, antes ou depois dos limitantes dessa classe são ditos **valores limites**.



14

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.1. INTRODUÇÃO (cont)

- ✔ Segundo alguns autores, a chance de se encontrar problemas em um software é maior se forem analisados os valores limites das partições.
 - ✔ Dessa forma surgiu a técnica denominada de **análise do valor limite** que é complementar à técnica de partição de equivalência e que é utilizada na especificação dos casos de teste funcionais de um sistema.
-

15

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.2. OBJETIVO DA TÉCNICA

- ✔ Trata-se de uma técnica de escolha seletiva de dados de teste das partições de equivalência.
 - ✔ A análise do valor limite prescreve que, ao se especificar casos de teste deve-se selecionar dados que estejam sobre os limites das classes de equivalência, que estejam localizados imediatamente antes de tais limites e imediatamente depois de tais limites.
-

16

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.3. DIRETRIZES

- ❓ Como escolher os valores limites quando se deseja especificar os casos de teste??
- ✅ Apesar de não existirem (até o momento) regras gerais aceita unanimemente na comunidade de computação, algumas regras práticas podem ser citadas!



17

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.3. DIRETRIZES

- 👉 Se a condição de entrada especifica um intervalo de valores válidos, devem ser selecionados dados para os limites dos valores e os dados das classes vizinhas que sejam imediatamente subsequentes a esse intervalo.
- Ex: [-1.00; +1.00]
- Usar: {-1.01, -1.00, +1.00, +1.01}



18

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.3. DIRETRIZES

👉 Se a condição de entrada especifica uma quantidade de valores, devem ser definidos dados de teste com uma quantidade inferior à mínima, igual à mínima, igual à superior e uma quantidade superior.

- Ex: De 1 a 255 entradas.
Usar: {nenhuma, 1, 255 e 256}



19

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.3. DIRETRIZES

👉 Aplicar as diretrizes anteriores considerando também os possíveis valores de saída para o sistema. Nesse caso, utilizar também os valores de saída como critério de definição das partições (ou subpartições).



20

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.3. DIRETRIZES

👍 Caso o programa trabalhe com a manipulação de arquivos (ex. envio e recebimento), selecionar casos de teste contendo arquivos vazios e arquivos que sejam iguais ou maiores do que as capacidades máximas de armazenamento ou envio especificadas.



21

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.3. DIRETRIZES

👍 Se a entrada ou a saída forem valores ordenados, dar maior atenção ao primeiro e ao último elemento.

Ex: lista_frutas = {"abacate",
"abricó", "manga", "pera", "uva"}

Usar: {"abacate", "uva"}



22

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.3. DIRETRIZES

👉 Se a entrada ou a saída forem valores ordenados, dar maior atenção ao primeiro e ao último elemento.

Ex: lista_frutas = {"abacate",
"abricó", "manga", "pera", "uva"}

Usar: {"abacate", "uva"}



23

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.4. CASOS ESPECIAIS

- ⚠️ O zero sempre deverá ser testado se estiver na partição de equivalência válida;
- ⚠️ Utilizar 0 e 59 quando estiver testando campos de minuto e segundo.
- ⚠️ Utilizar 0 e 23 para os campos de hora.
- ⚠️ Atentar para os dias dos meses em especial nos anos bissextos.



24

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.5. EXEMPLO PRÁTICO / EXERCÍCIO

1) Um programa é utilizado para o cálculo do valor de aluguel a ser pago de acordo com a seguinte tabela (valor do aluguel nominal: R\$ 500,00).

Data de Pagamento	Cálculo do valor pago
Até o 5º Dia do mês	10% de desconto sobre o valor nominal
Até o 10º dia do mês	5% de desconto sobre o valor nominal
Até o 15º dia do mês	Valor nominal
Após o 15º dia do mês	Multa de 2% mais juros de 1,5% ao mês (0,05% ao dia).

25

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

3.5. EXEMPLO PRÁTICO / EXERCÍCIO (cont)

Especifique um conjunto de casos de teste que seja adequado considerando o critério de análise de valores limites. Execute usando o JUNIT 5, mostrando o relatório de teste e o código de teste.

Além disso, utilize os casos de teste especificados para testar a biblioteca fornecida pelo professor.

Obs: Caso o dia do mês seja menor do que um ou maior do que 31, o componente retorna -1.

26

3. CRITÉRIO DE ANÁLISE DE VALORES LIMITES

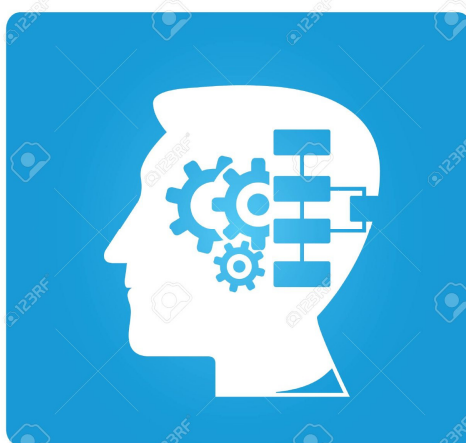
3.5. EXEMPLO PRÁTICO / EXERCÍCIO (cont)

2) Considere uma função f que aceita como entradas de 4 a 6 valores inteiros de 2 dígitos cada. Aplique a técnica de análise do valor limite e defina os casos de teste para a função.

Considere que os valores são informados separando-se cada um deles por vírgulas e que os espaços em branco e zeros à esquerda são ignorados.

27

4. TESTE FUNCIONAL SISTEMÁTICO



28

4. TESTE FUNCIONAL SISTEMÁTICO

4.1. DEFINIÇÃO

- ✔ O critério de teste funcional sistemático é uma composição dos critérios de partição de equivalência e análise dos valores limites.
- ✔ Este critério essencialmente prescreve diretivas adicionais para a determinação dos dados de teste, considerando os domínios de entrada e saída.

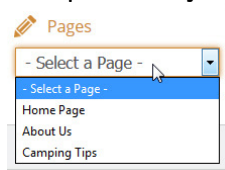
29

4. TESTE FUNCIONAL SISTEMÁTICO



4.2. CRITÉRIOS (“sistemáticos”)

- ✔ Caso o domínio de entrada apresente valores discretos facilmente enumeráveis, testar todos os valores.

Exemplo: Se uma entrada for um “select” em um documento HTML, testar todos os valores possíveis para seleção da entrada.



30





4. TESTE FUNCIONAL SISTEMÁTICO



4.2. CRITÉRIOS (“sistemáticos”)

✔ Caso o domínio de entrada seja um intervalo de valores, utilizar os critérios de análise de valores limites e, adicionalmente, utilizar ao menos um dado no interior do intervalo.

Exemplo: Domínio de entrada [1, 10].
Dados de teste: {0.99; 1; 1.01, 5.00, 10, 10.99}



31





4. TESTE FUNCIONAL SISTEMÁTICO

4.2. CRITÉRIOS (“sistemáticos”)

✔ Testar valores ilegais para verificar se o software os rejeita.

Exemplo: Entrada que aceita uma string que representa um e-mail.
Dados de teste ilegais possíveis:
{“@@@”, “ ”, “ @”, “abcdef”, “!”}



32

4. TESTE FUNCIONAL SISTEMÁTICO

4.2. CRITÉRIOS (“sistemáticos”)

- ✔ Caso o intervalo de valores de entrada seja variável (e.g.: dependente de outra variável de entrada), testar as combinações relacionais possíveis.

Exemplo: $x \in [0,1]$ e $y \in [x, 2x]$.

Casos de teste possíveis:

$\langle 0, 0 \rangle$, “válido”; $\langle 1, 2 \rangle$, “válido”
 $\langle 1, 3 \rangle$, “inválido”; $\langle 1, 0 \rangle$, “inválido”
 $\langle -1, -1 \rangle$, “inválido”; $\langle 2, 1 \rangle$, “inválido”
 $\langle 2, 4 \rangle$, “inválido”; $\langle 1, 2.01 \rangle$, “inválido”



33

4. TESTE FUNCIONAL SISTEMÁTICO

4.2. CRITÉRIOS (“sistemáticos”)

- ✔ Caso o intervalo de valores de entrada seja variável (e.g.: dependente de outra variável de entrada), testar as combinações relacionais possíveis.

Exemplo: $x \in [0,1]$ e $y \in [x, 2x]$.

Casos de teste possíveis:

$\langle 0, 0 \rangle$, “válido”; $\langle 1, 2 \rangle$, “válido”
 $\langle 1, 3 \rangle$, “inválido”; $\langle 1, 0 \rangle$, “inválido”
 $\langle -1, -1 \rangle$, “inválido”; $\langle 2, 1 \rangle$, “inválido”
 $\langle 2, 4 \rangle$, “inválido”; $\langle 1, 2.01 \rangle$, “inválido”



34

5. GRAFOS CAUSA-EFEITO

5.1. DEFINIÇÃO

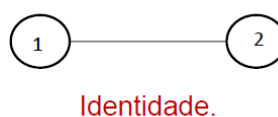
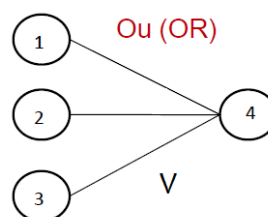
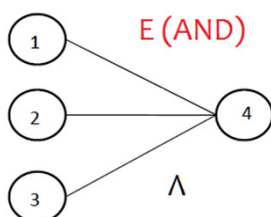
Um grafo causa-efeito representa um programa sob o ponto de vista de combinações de suas entradas (causa) e possíveis saídas (efeito).

Nessa técnica, os dados de teste são selecionados tomando-se como base os valores possíveis para tais combinações.

35

5. GRAFOS CAUSA-EFEITO

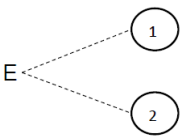
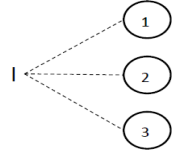
5.2. NOTAÇÃO (CAUSA-EFEITO)



36

5. GRAFOS CAUSA-EFEITO

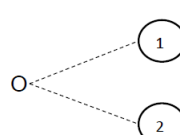
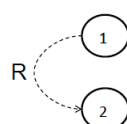
5.2. NOTAÇÃO (RESTRIÇÕES)

No máximo uma das causas pode ser verdadeira.	
No mínimo uma das causas deve ser verdadeira, podendo todas elas serem verdadeiras simultaneamente.	

37

5. GRAFOS CAUSA-EFEITO

5.2. NOTAÇÃO (RESTRIÇÕES)

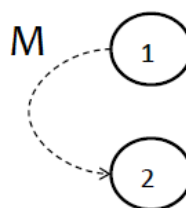
Uma e somente uma das causas deve ser verdadeira.	
Se uma das causas é verdadeira, então a outra é falsa.	

38

5. GRAFOS CAUSA-EFEITO

5.2. NOTAÇÃO (RESTRIÇÕES)

Se uma das causas é verdadeira, então a outra causa é forçada a ser falsa.



39

5. GRAFOS CAUSA-EFEITO

5.3. UTILIZAÇÃO DA TÉCNICA

Para se utilizar o critério baseado em grafos causa e efeito deve-se:

1. Desenhar o grafo correspondente ao programa considerando causas e efeitos;
2. Montar uma tabela booleana associando-se as causas aos efeitos;
3. Especificar um caso de teste para cada entrada da tabela.

40

5. GRAFOS CAUSA-EFEITO

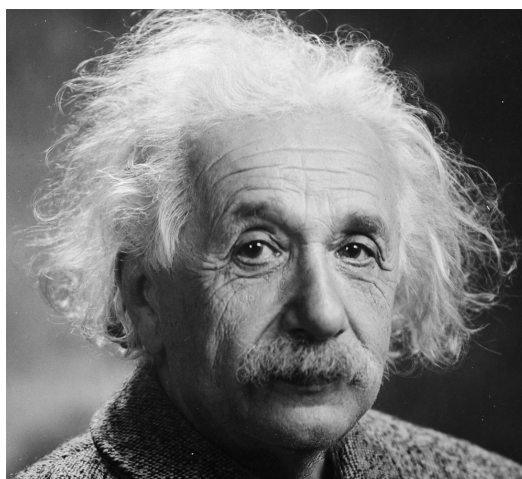
5.4. EXEMPLO:

Um programa lê dois caracteres e imprime mensagens de acordo com o seguinte. Caso o primeiro caractere seja A ou B e o segundo seja um dígito, o programa imprime “OK”. Caso o primeiro caractere seja incorreto, imprime “Erro 1”. Caso o segundo caractere seja incorreto, imprime “Erro 2”.

Especificar casos de teste considerando grafos causa-efeito montando uma **tabela de decisão**.

41

6. ERROR GUESSING



42

6. ERROR GUESSING

6.1. DEFINIÇÃO

Essa técnica se baseia na especificação de casos de teste de acordo com a experiência do testador.

Valores são selecionados de acordo com a experiência acumulada em se testar um software de uma dada categoria (valores problemáticos ou que costuma revelar defeitos mas que não são prescritos explicitamente por nenhum critério).

43

7. TABELA DE DECISÃO

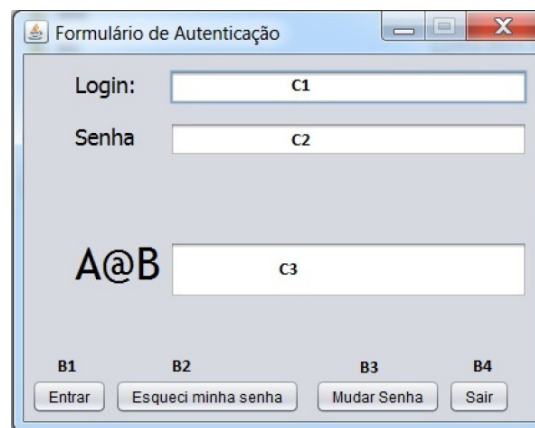
7.1. DEFINIÇÃO E OBJETIVOS

- ✔ Uma tabela de decisão representa combinações de entradas de um sistema e os efeitos esperados para suas saídas, de acordo com as regras do sistema.
 - ✔ O critério de teste baseado em tabela de decisão prescreve a especificação de um caso de teste para cada uma das regras do sistema.
-

44

7. TABELA DE DECISÃO

7.2. EXEMPLO PRÁTICO



45

7. TABELA DE DECISÃO

7.2. EXEMPLO PRÁTICO

Entrada/Regra	R0	R1	R2	R3	R4
C1	F	V	V	V	X
C2	F	V	V	V	X
C3	F	V	V	V	X
B1	F	V	F	F	X
B2	F	F	V	F	X
B3	F	F	F	V	X
B4	F	F	F	F	V
Ação	NDA	OK	OK	OK	OK
Obs:	Estado inicial do aplicativo.	Acesso correto	Lembrança gerada	Mudança efetuada	Fechar aplicação

46

7. TABELA DE DECISÃO

7.2. EXEMPLO PRÁTICO

Caso de teste (baseado na regra R1) :

<["admin", "admin", "A@B", "click em B1],
"Login efetuado com sucesso">

47

8. MATRIZ DE RASTREABILIDADE

8.1. DEFININDO RASTREABILIDADE

Na área de desenvolvimento de sistemas, rastreabilidade significa comumente relacionamento entre diferentes artefatos produzidos ao longo do ciclo de desenvolvimento.

A rastreabilidade é útil tanto para as atividades de planejamento de desenvolvimento (incluindo planejamento de testes) como para as próprias atividades do ciclo em si (execução de diferentes tarefas de desenvolvimento).

48

8. MATRIZ DE RASTREABILIDADE

8.2. RASTREABILIDADE REQUISITO – TESTE

A rastreabilidade entre requisitos de sistema (ou módulo/componente) é tipicamente expressa por uma matriz (tabela) que relaciona cada requisito a um teste que é o responsável pela sua validação e verificação.

A **matriz de rastreabilidade** fornece uma visão geral da **cobertura dos casos de teste em termos de requisitos funcionais**.

É comum utilizar uma coluna de controle correspondente à aprovação ou reprovação do requisito pelo caso de teste.

49

8. MATRIZ DE RASTREABILIDADE

8.2. RASTREABILIDADE REQUISITO – TESTE

ID DO REQUISITO	ID DO CASO DE TESTE	RESULTADO
R.1	CT.1	APROV/REPROV
R.2	CT.1.; CT.2	APROV/REPROV
...
R.N	CT. M	APROV/REPROV

Notar que um requisito (R.i) pode ser diretamente verificado por mais de um caso de teste (CT.j).

50

