

# Teste de Software



Prof. Marcos Rodrigo momo, M.Sc.  
[marcos.momo@ifsc.edu.br](mailto:marcos.momo@ifsc.edu.br)

Gaspar, maio 2020.

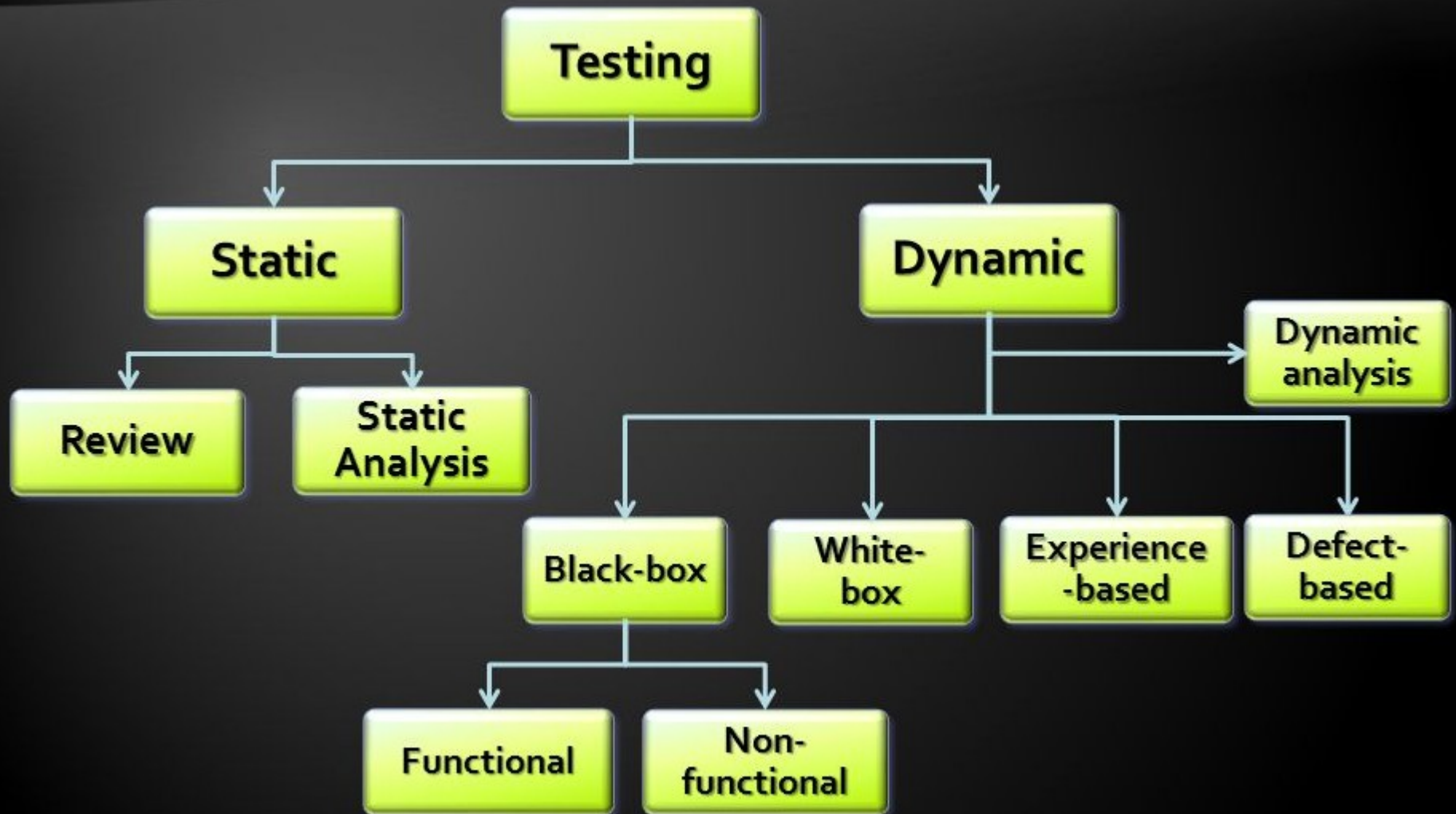



# Roteiro

---

- **Teste funcional**
- Teste estrutural
- Técnica estático

# Visão geral das técnicas de teste de software





# Classificação das técnicas de teste existentes

---

- **Teste Funcional ou de caixa-preta**
  - Também chamadas de técnicas baseadas em especificação, são uma forma de derivar e selecionar as condições e casos de testes **baseados na análise da documentação**.
  - Isto inclui testes funcionais e não funcionais, para um componente ou sistema **sem levar em consideração a sua estrutura interna**.
- **Teste Estrutural ou de caixa-branca**
  - São baseadas na estrutura interna de um componente ou sistema.
- **Teste Baseado na experiência**
  - São testes **derivados da intuição e conhecimento dos testadores** através de sua experiência em aplicações e tecnologia similares.

# Como escolher a técnica de teste a ser usada?

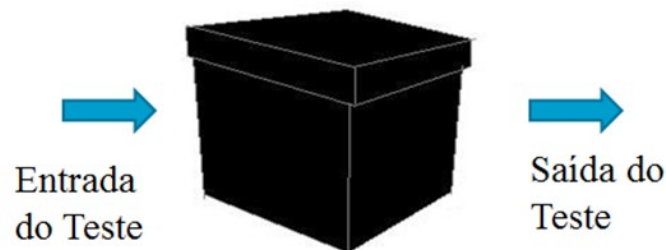
---

- **A escolha de qual técnica utilizar dependerá de uma série de fatores, incluindo:**
  - Tipo de sistema
  - Padrões
  - Requisitos contratuais
  - Nível do risco e tipos de riscos
  - Documentação disponível
  - Conhecimento dos testadores
  - Tempo e dinheiro
  - Ciclo de desenvolvimento etc
- Algumas técnicas são mais facilmente aplicadas em certas situações e níveis de teste, já outras são aplicáveis a todos os níveis.
- Ao criar casos de teste, os testadores geralmente usam uma combinação de técnicas de teste para garantir uma **cobertura adequada do objeto em teste.**

# Teste funcional

---

- i Teste funcional é uma técnica utilizada para se projetarem casos de teste na qual o programa ou sistema é considerado uma **caixa preta**.
- i Para testá-lo, são fornecidas entradas e avaliadas as saídas geradas para verificar **se estão em conformidade com os objetivos especificados**.
- i Nessa técnica os **detalhes de implementação não são considerados** e o *software* é avaliado segundo o ponto de vista do usuário.



Requer boa especificação dos requisitos.



Aplicada em produtos criados em diversos paradigmas.

# Analogia do teste funcional



Lote, terreno para a construção de uma casa



Equipe que vai construir a casa



Questões  
externas





# Teste funcional

---

- i O teste funcional pode detectar todos os defeitos, submetendo o programa ou sistema a todas as possíveis entradas (**Teste Exaustivo**).
- i No entanto, o domínio de entrada pode ser infinito ou muito grande, de modo a tornar o tempo da atividade de teste inviável, fazendo com que essa alternativa se torne impraticável.



# Teste funcional

---

- i Essa limitação da atividade de teste, **que não permite afirmar que o programa esteja correto**, fez com que fossem definidas as **técnicas de teste** e os **diversos critérios** pertencentes a cada uma delas.
- i Assim, é possível conduzir essa atividade de maneira mais **sistemática**, podendo-se inclusive, dependendo do critério utilizado, ter uma avaliação quantitativa da atividade de teste.



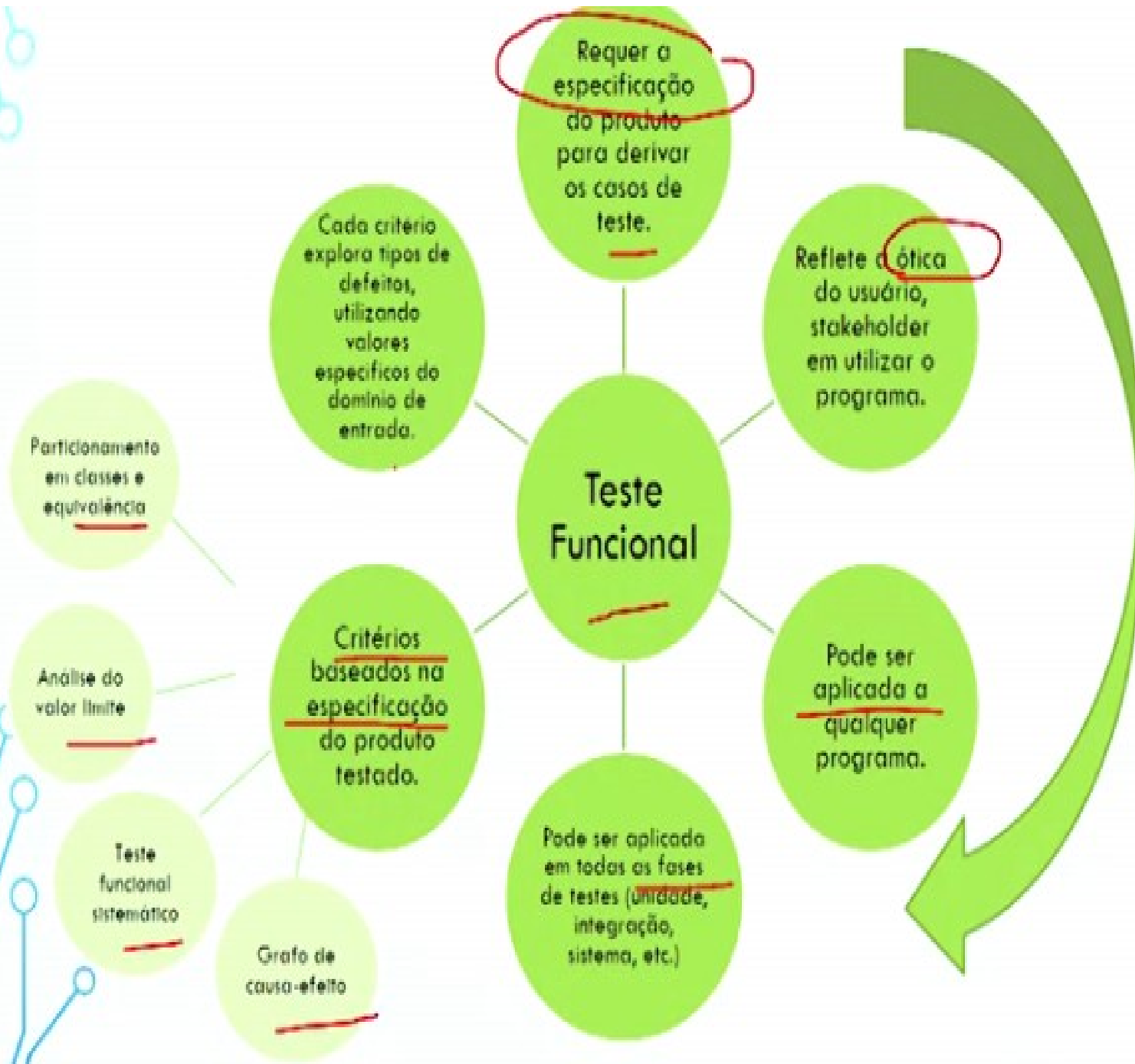
# Teste funcional

---

- i O que distingue essencialmente as três técnicas de teste – **Funcional, Estrutural e Baseada na experiência** – é a fonte utilizada para definir os requisitos de teste.
- i Além disso, **cada critério de teste procura explorar determinados tipos de defeitos**, estabelecendo requisitos de teste para os quais valores específicos do domínio de entrada do programa devem ser definidos com o intuito de exercitá-los.

# Teste funcional

- Este teste considera a especificação do software em si e não no código
- Pode ser aplicada em todas as fases de testes unitário, integração, sistemas entre outros
- Pode ser aplicado por equipes independentes
- Pode ser aplicado a produtos desenvolvidos em diversos paradigmas e linguagens de programação





# Critérios de teste funcional

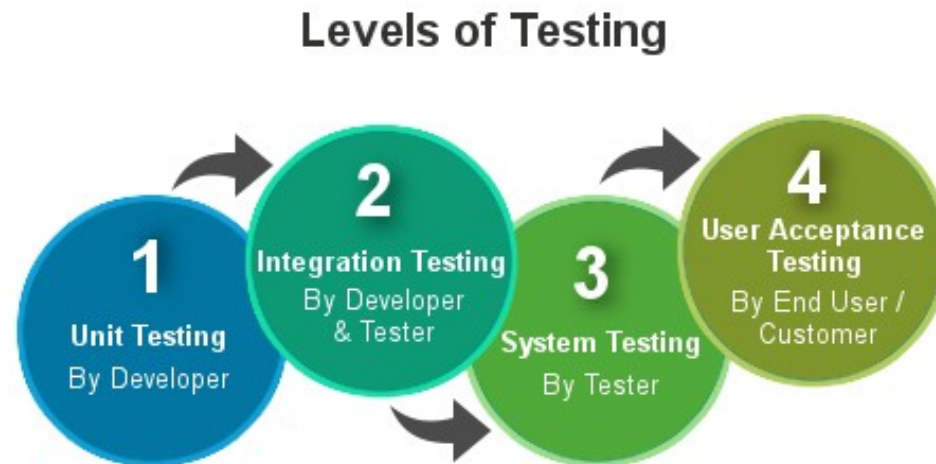
---

- i Como todos os critérios da técnica funcional baseiam-se **apenas na especificação do produto testado**, a qualidade de tais critérios depende fortemente da existência de uma **boa especificação de requisitos**.
- i Especificações ausentes ou mesmo incompletas tornarão difícil a aplicação dos critérios funcionais.
- i Além disso, tais critérios também apresentam a limitação de não garantir que partes essenciais ou críticas do produto em teste sejam exercitadas.

# Critérios de teste funcional

---

- i Os critérios funcionais podem ser aplicados em **todas as fases/níveis de testes** e em produtos desenvolvidos com **qualquer paradigma de programação**, pois não levam em consideração detalhes de implementação.





# Critérios de teste funcional

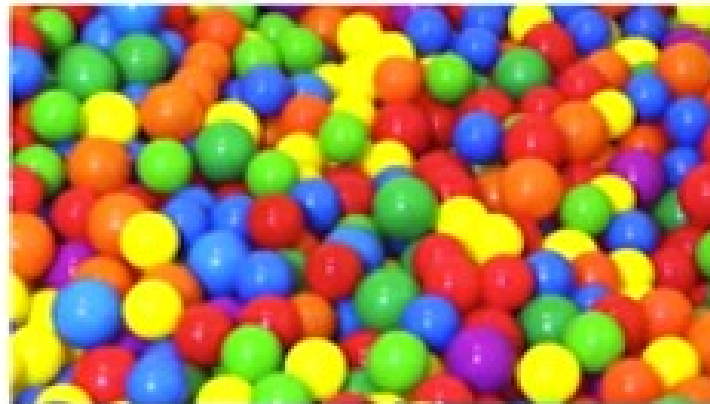
---

- i Os critérios mais conhecidos da técnica de teste funcional são:
  - | Partição de Equivalência
  - | Análise do Valor Limite
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Grafo Causa-Efeito
  - | Error Guessing

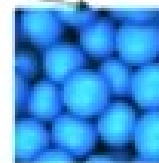
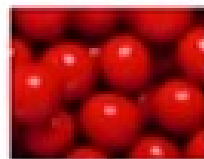
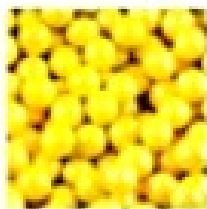


# Partição de Equivalência

Domínio de entrada  
(infinito ou muito grande)




Subconjuntos do domínio



# Particionamento em classes de equivalência

---


- i Considerando-se que o Teste Exaustivo é, em geral, impossível de ser aplicado, durante a atividade de teste, o testador fica restrito a usar um subconjunto de todas as possíveis entradas do programa, sendo que **esse subconjunto deve ter uma grande probabilidade de encontrar defeitos**.
- i Com o objetivo de apoiar a determinação desse subconjunto, tornando a quantidade de dados de entrada **finita** e mais **viável** para uma atividade de teste, esta técnica **divide o domínio de entrada em classes de equivalência** que, de acordo com a especificação do programa, são tratadas da mesma maneira.
- i Uma vez definidas as classes de equivalência, pode-se assumir com alguma segurança que **qualquer elemento da classe pode ser considerado um representante desta, pois todos eles devem se comportar de forma similar**.



# Particionamento em classes de equivalência

---

- i Ou seja, **se um elemento de uma classe detectar um defeito, qualquer outro também detecta;**
- i Se não detectar, presume-se que os outros também não detectam.
- i Assim, o critério **reduz o domínio de entrada a um tamanho possível** de ser tratado durante a atividade de teste.
- i Além do domínio de entrada, alguns autores consideram também o domínio de saída, identificando neste possíveis alternativas que poderiam determinar classes de equivalência no domínio de entrada.



# Particionamento em classes de equivalência

---

- i Para ajudar na identificação das partições, pode-se observar a especificação do *software* procurando termos como “intervalo” e “conjunto” ou palavras similares que indiquem que os dados são processados da mesma forma.
- i Uma **classe de equivalência** representa um conjunto de estados **válidos** ou **inválidos** para as condições de entrada.



# Critérios de teste funcional

---

- i Os critérios mais conhecidos da técnica de teste funcional são:
  - | **Partição de Equivalência**
  - | Análise do Valor Limite
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Grafo Causa-Efeito
  - | Error Guessing



# Critérios de teste funcional

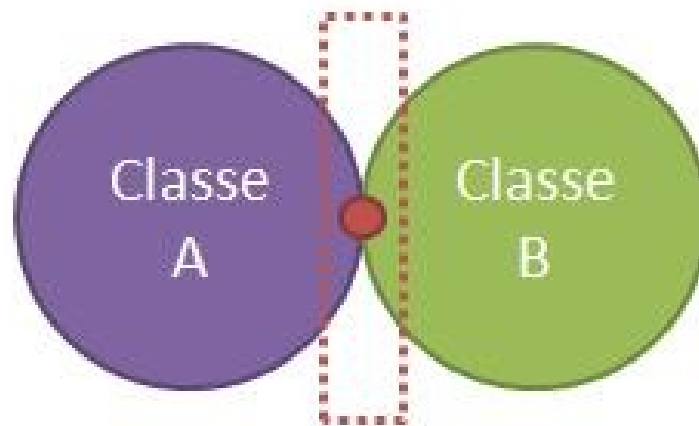
---

- i Os critérios mais conhecidos da técnica de teste funcional são:
  - | **Partição de Equivalência**
  - | Análise do Valor Limite
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Grafo Causa-Efeito
  - | Error Guessing

# Análise do Valor Limite

---

Análise de Valor Limite





# Análise do Valor Limite ou *Boundary Value Analysis (BVA)*

---

- i Um grande número de erros ocorre **nas fronteiras do domínio de entrada em vez de no “centro”**.
- i A experiência mostra que casos de teste que exploram condições limites têm uma maior probabilidade de encontrar defeitos.
- i Tais condições correspondem a valores que estão **exatamente sobre ou imediatamente acima ou abaixo dos limitantes das classes de equivalência**.
- i Assim, esse critério é usado em **conjunto** com o Particionamento de Equivalência, mas em vez de os dados de teste serem escolhidos aleatoriamente, eles devem ser selecionados de forma que o limitante de cada classe de equivalência seja explorado (seleciona-se os casos de teste nas “bordas” das classes).



# Análise do Valor Limite ou *Boundary Value Analysis (BVA)*

---

- i Além da escolha seletiva dos dados de testes, outro ponto que distingue esse critério do Particionamento de Equivalência é a **observação do domínio de saída**.
- i Embora não existam diretrizes bem definidas que levem à determinação dos dados de teste, as seguintes recomendações são sugeridas:
  1. Se a condição de entrada especifica um intervalo de valores, devem ser definidos dados de teste para os **limites** desse intervalo e dados de teste **imediatamente subsequentes**, que explorem as classes inválidas vizinhas desse intervalo.

Ex.: se uma classes válida estiver no intervalo -1,0 e +1,0, devem ser definidos os seguintes dados de teste: -1,0; +1,0; -1,001 e +1,001;



# Análise do Valor Limite ou *Boundary Value Analysis (BVA)*

---

2. Se a condição de entrada especifica uma **quantidade** de valores, por exemplo, de 1 a 255 valores, devem ser definidos dados de teste com **nenhum** valor de entrada, **somente um** valor de entrada, **255** valores e **256** valores de entrada;
3. Usar a recomendação 1 e 2 para as condições de saída. Por exemplo, considere que uma tabela de temperatura \* pressão é esperada como saída de um programa de análise de engenharia. Casos de teste devem ser projetados para criar um relatório de saída que produza o número máximo (e mínimo) admissível de entradas na tabela;
4. Se a entrada ou saída for um conjunto ordenado, deve ser dada maior atenção aos primeiro e último elementos desse conjunto;
5. Usar a intuição para definir outras condições limite.



# Critérios de teste funcional

---

- | Os critérios mais conhecidos da técnica de teste funcional são:
  - | Partição de Equivalência
  - | Análise do Valor Limite
  - | Sistemático
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Grafo Causa-Efeito
  - | Error Guessing



# Teste funcional sistemático

---

- i Esse critério combina os critérios Particionamento de Equivalência e Análise do Valor Limite.
- i 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 que defeitos coincidentes mascarem falhas.
- i Além disso, ele também requer a avaliação no limite de cada partição e subsequente a ele.

# Teste funcional sistemático

## Diretrizes: 1 - Valores Numéricos

---

- i Para facilitar a identificação desses casos de teste, o critério fornece as seguintes diretrizes:
  - i Para **o domínio de entrada**, selecionar valores de entrada da seguinte forma:
    - (a) Valores discretos: testar todos os valores; e
    - (b) Intervalo de valores: testar os extremos e um valor no interior do intervalo.
  - i **Para o domínio de saída**, selecionar valores de entrada que resultem nos valores de saída que devem ser gerados pelo software.
  - i Os tipos das saídas podem não coincidir com os tipos dos valores de entrada; por exemplo, valores de entrada distintos podem produzir um intervalo de valores de saída, dependendo de outros fatores, ou um intervalo de valores de entrada pode produzir somente um ou dois valores de saída como verdadeiro e falso.
  - i Assim, devem-se escolher como entrada valores que explorem os valores de saída da seguinte forma:
    - (a) Valores discretos: gerar cada um deles; e
    - (b) Intervalo de valores: gerar cada um dos extremos e ao menos um valor no interior do intervalo.

# Teste funcional sistemático

Diretrizes: 2 – Tipos de valores diferentes e casos especiais

---

- i Tipos diferentes de valores devem ser explorados na entrada e na saída como, por exemplo, espaço em branco, que pode ser interpretado como zero em um campo numérico.
- i Casos especiais como zero sempre devem ser selecionados individualmente, mesmo que ele esteja dentro de um intervalo de valores.
- i O mesmo serve para valores nos limites da representação binária dos dados.



# Teste funcional sistemático

Diretrizes: 3 – Valores Ilegais

---

- Valores que correspondem a entradas ilegais também devem ser incluídos nos casos de teste, para assegurar que o software os rejeita.
- Deve-se também tentar gerar valores ilegais, os a quais não devem ser bem-sucedidos.
- É importante que sejam selecionados os limites dos intervalos numéricos, tanto inferior quanto superior, valores imediatamente fora dos limites desses intervalos e também os valores imediatamente subsequentes aos limites do intervalo e pertencentes a ele.



# Teste funcional sistemático

## Diretrizes: 4 – Números reais

---

- i Valores reais envolvem mais problemas do que valores inteiros, uma vez que, na entrada, são fornecidos como números decimais, para processamento, são armazenados na forma binária e, como saída, são convertidos em decimais novamente.
- i Verificar o limite para números reais pode não ser exato, mas, ainda assim, essa verificação deve ser incluída como caso de teste.
- i Deve ser definida uma margem de erro de tal forma que, se ultrapassada, então o valor pode ser considerado distinto.
- i Além disso, devem ser selecionados números reais bem pequenos e também o zero.



# Teste funcional sistemático

## Diretrizes: 5 – Intervalos variáveis

---

- i Ocorre quando o intervalo de uma variável depende do valor de outra variável.
- i Por exemplo, suponha que o valor da variável  $x$  pode variar de zero ao valor da variável de  $y$  e que  $y$  é um inteiro positivo.
- i Nessa caso, os seguintes dados de entrada devem ser definidos:
  - i.  $x = y = 0$ ;
  - ii.  $x = 0 < y$ ;
  - iii.  $0 < x = y$ ;
  - iv.  $0 < x < y$ ;
- i Além disso, devem-se também selecionar os seguintes valores ilegais:
  - i.  $y = 0 < x$ ;
  - ii.  $0 < y < x$ ;
  - iii.  $x < 0$ ;
  - iv.  $y < 0$ ;



# Teste funcional sistemático

## Diretrizes: 6 – Arranjos

---

- i Quando se usa arranjo, tanto como entrada como saída, deve-se considerar o fato de o tamanho do arranjo ser variável bem como de os dados serem variáveis.
- i Os elementos do arranjo devem ser testados como se fossem variáveis comuns, como mencionado no item anterior.
- i Além disso, o tamanho do arranjo deve ser testado com valores intermediários, com os valores mínimo e máximo.
- i Para simplificar o teste, podem-se considerar as linhas e colunas de um arranjo como se fossem subestruturas, a serem testadas em separado.
- i Assim, o arranjo deve ser testado primeiro como uma única estrutura, depois como uma coleção de subestruturas, e cada subestrutura deve ser testada independentemente.



# Teste funcional sistemático

Diretrizes: 7 – Dados tipo texto ou string

---

- i Neste caso o dado de entrada deve explorar comprimentos variáveis e também validar os caracteres que o compõem, pois algumas vezes os dados de entrada podem ser apenas alfabéticos, outras vezes, alfanuméricos, e algumas vezes podem possuir caracteres especiais.

# Teste funcional sistemático

---

- i Além desses casos, o critério Funcional Sistemático requer que dois elementos de uma mesma classe de equivalência sejam explorados para evitar erros do seguinte tipo:
  - | Suponha que o programa recebe um valor de entrada e produza seu quadrado.
  - | Se o valor de entrada for 2, e o valor produzido como saída for 4, embora o resultado esteja correto, ainda assim não se pode afirmar qual das operações o programa realizou, pois poderia ser  $2*2$  como também  $2+2$ .



# Critérios de teste funcional

---

- | Os critérios mais conhecidos da técnica de teste funcional são:
  - | Partição de Equivalência
  - | Análise do Valor Limite
  - | Sistemático
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Grafo Causa-Efeito
  - | Error Guessing



# Tabelas de Decisão

---

- i As especificações geralmente contêm regras de negócio que definem o **funcionamento do sistema e as condições em que cada função opera.**
- i **Decisões individuais são normalmente simples, mas o efeito geral dessas condições lógicas pode ser um tanto quanto complexo.**
- i **Como testador, é necessário garantir que cada combinação em que essas condições ocorrem sejam testadas.**
- i As tabelas de decisões são, portanto, um mecanismo para capturarem todas as decisões lógicas de um sistema.

# Tabelas de Decisão

- i A tabela de decisão deve ter uma estrutura que lista todas as condições de entrada e todas as ações que podem surgir a partir delas, e estão estruturadas de forma que as linhas representam as condições e suas possíveis ações.
- i As regras de negócio que envolvem combinações de condições para produzir um conjunto de ações estão organizadas no topo.
- i Portanto, cada coluna representa um possível caso de teste, haja vista que ele aborda as entradas e saídas esperadas.



# Tabelas de Decisão

	Regra de Negócio 1	Regra de Negócio 2	Regra de Negócio 3
Condição 1	V	F	V
Condição 2	V	V	V
Condição 3	V	-	F
Ação 1	SIM	NÃO	SIM
Ação 2	NÃO	SIM	SIM

- i O número de condições e ações pode ser um tanto quanto alto, mas geralmente o número de combinações que produz uma ação é relativamente pequeno. Por esta razão, não é necessário entrar com todas as combinações de condições dentro da tabela de decisão, mas restringi-las às combinações que correspondem a regras de negócio.



# Tabelas de Decisão

- i Como exemplo, podemos citar o seguinte cenário:
- “Um supermercado tem um programa de fidelidade oferecido para todos os clientes. Os clientes que possuem o cartão de fidelidade gozam do benefício de desconto adicional em todas as compras (regra 3) ou acumulação de pontos de fidelidade (regra 4), que podem ser convertidos em vouchers para o supermercado ou pontos equivalentes na rede de parceiros. Clientes sem o cartão de fidelidade recebem um desconto adicional somente se gastarem mais de R\$100,00 em qualquer visita à loja (regra 2), caso contrário, somente as promoções ofertadas a todos os clientes se aplicam.”

	Regra 1	Regra 2	Regra 3	Regra 4
Condições				
Cliente com cartão fidelidade	V	V	F	F
Cliente sem Cartão fidelidade	F	F	V	V
Desconto Extra selecionado	-	-	V	F
Gasto > R\$100	F	V	-	-
Ações				
Sem desconto	V	F	F	F
Desconto Extra	F	V	V	F
Pontos de fidelidade	F	F	F	T

# Tabelas de Decisão

- i A partir da tabela de decisão pode-se determinar casos de teste através do ajuste dos valores para as condições e determinação da saída esperada.
- i Por exemplo, pela regra 1 podemos ter um cliente normal com uma transação de R\$50,00 e checar se nenhum desconto foi aplicado.
- i Caso o mesmo cliente realize uma operação de R\$150,00 um desconto deve ser aplicado.
- i Dessa forma, cada coluna da tabela de decisão representa um caso de teste.

	Regra 1	Regra 2	Regra 3	Regra 4
Condições				
Cliente com cartão fidelidade	V	V	F	F
Cliente sem Cartão fidelidade	F	F	V	V
Desconto Extra selecionado	-	-	V	F
Gasto > R\$100	F	V	-	-
Ações				
Sem desconto	V	F	F	F
Desconto Extra	F	V	V	F
Pontos de fidelidade	F	F	F	T



# Critérios de teste funcional

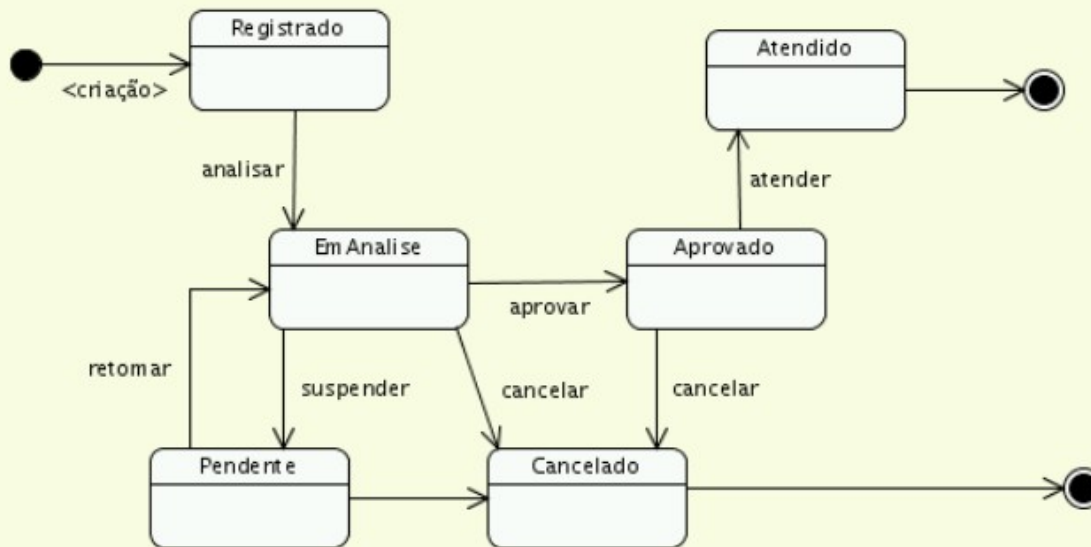
---

- | Os critérios mais conhecidos da técnica de teste funcional são:
  - | Partição de Equivalência
  - | Análise do Valor Limite
  - | Sistemático
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Grafo Causa-Efeito
  - | Error Guessing

# Teste de Transição de Estados

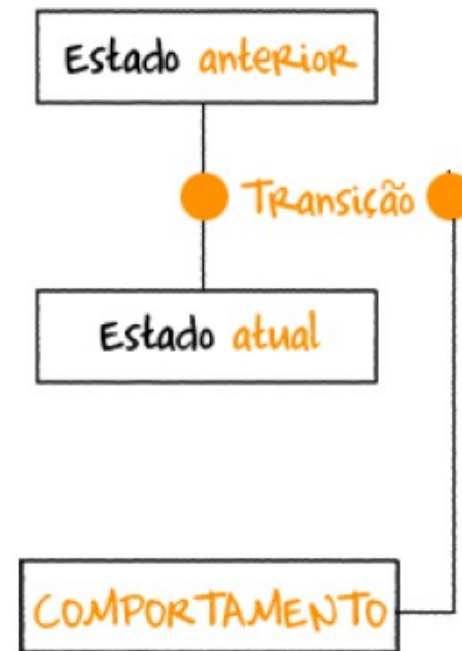
- Um sistema pode exibir respostas diferentes dependendo da sua condição atual ou de estado anterior podendo ser representado por um **diagrama de transição de estados (UML)**.
- Permite ao testador visualizar o *software* em termos de estados, transições entre estados, as entradas ou eventos que disparam as mudanças de estado (transição) e as ações que podem resultar daquelas transições.
- Uma **tabela de estado** exibe a **relação entre estados e entradas**, e pode destacar possíveis transições inválidas.

## Pedido de Compra



- Os testes podem ser construídos para cobrir:
  - uma sequência típica de *status*
  - cobrir todos os estados
  - exercitar todas as transições
  - exercitar uma sequência específica de transições
  - ou testar transições inválidas.

# Diagramas de Transição de Estado

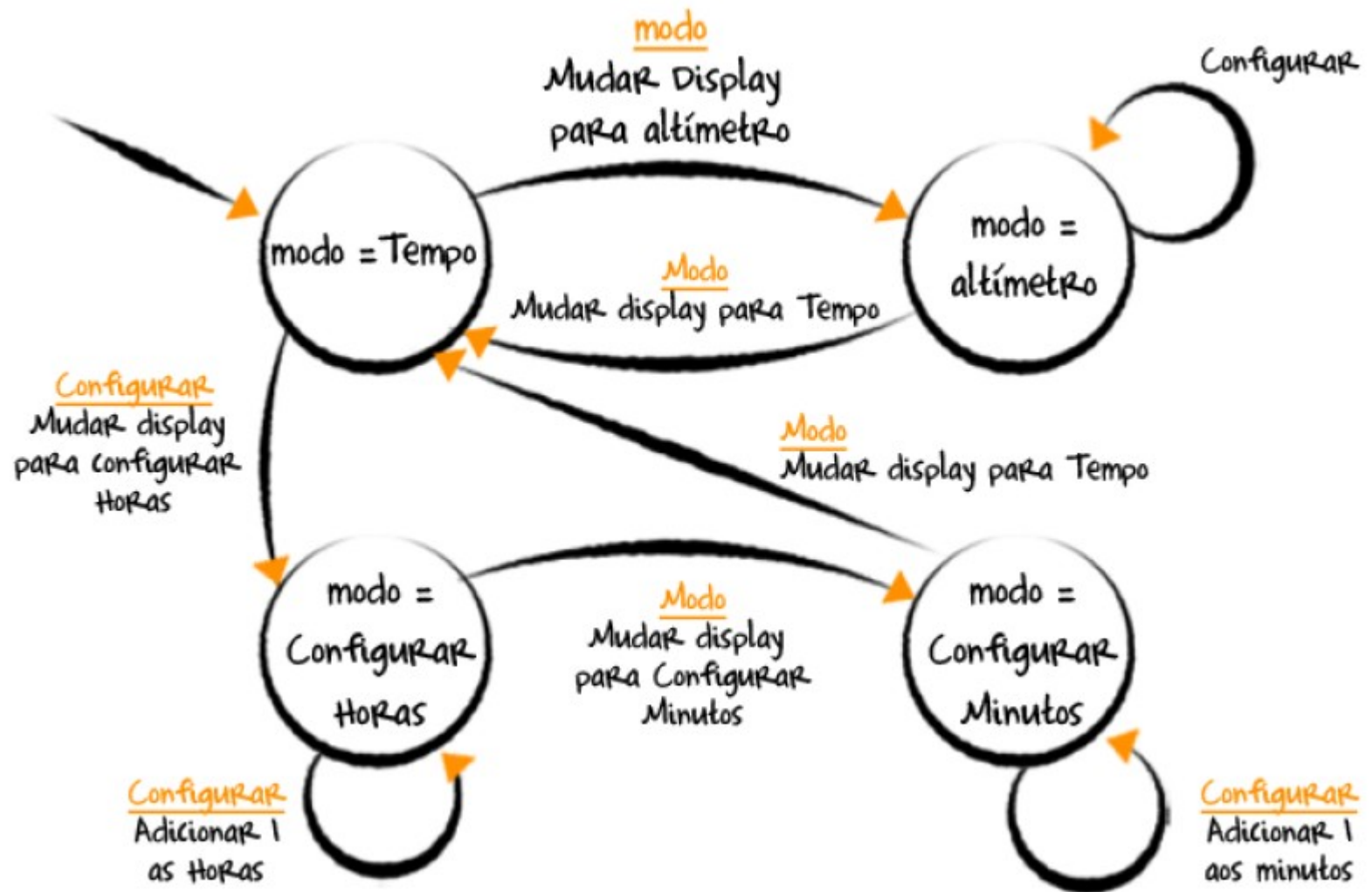


# Diagramas de Transição de Estado

---

- i Como exemplo, podemos citar o cenário abaixo:
  - | Um relógio de escalar tem 2 modos: Tempo e Altímetro. No modo Tempo, se pressionar “modo”, o relógio muda para o modo Altímetro e, se pressionado novamente, retorna para o modo Tempo. Enquanto o relógio estiver no modo Altímetro, se o botão SET for pressionado, nenhum efeito é gerado.
  - | Quando o relógio estiver no modo Tempo, se pressionar o botão SET, o relógio muda para SET HRS, e o display de tempo pode ser incrementado se o botão SET for pressionado. Se o botão Modo for pressionado enquanto o relógio estiver em SET HRS, o relógio muda para SET MIN, onde se o botão SET for pressionado, os minutos são incrementados. Se o botão modo for pressionado, o relógio retorna para o modo Tempo.
  - | Observe que nem todos os eventos têm um efeito em todos os estados. Onde o evento não tiver nenhum efeito, em um determinado estado é normalmente omitido, mas pode ser representado por uma seta que inicia e retorna para o mesmo estado, indicando que não cabem transições, o que também é conhecida como transação nula.

# Diagramas de Transição de Estado





# Critérios de teste funcional

---

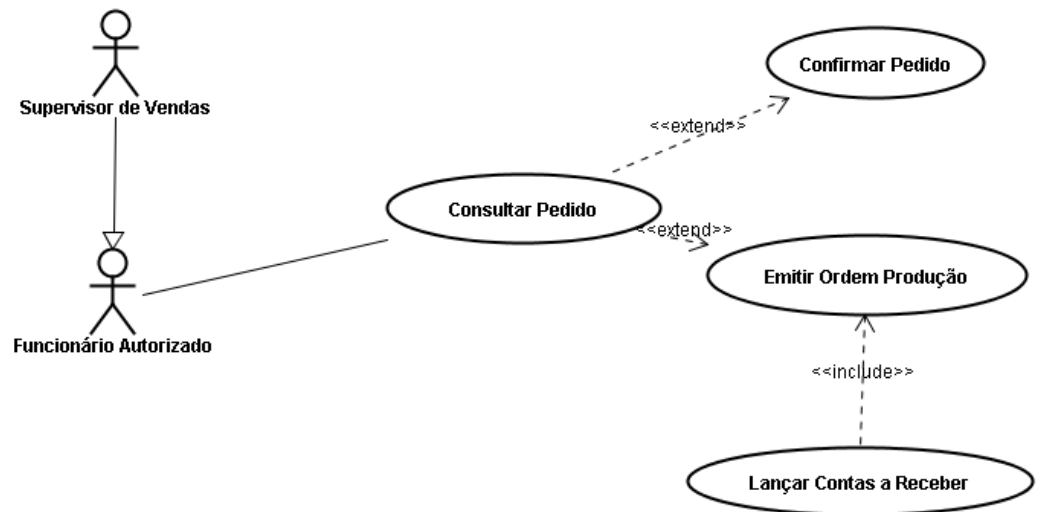
- | Os critérios mais conhecidos da técnica de teste funcional são:
  - | Partição de Equivalência
  - | Análise do Valor Limite
  - | Sistemático
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Grafo Causa-Efeito
  - | Error Guessing



# Teste de Caso de Uso

- Testes **podem ser especificados a partir de casos de uso** ou cenários de negócios.
- Caso de uso:
  - Descreve interações entre os atores (usuários e o sistema) que produz um resultado relevante para um usuário do sistema.
  - Tem pré-condições, que precisam ser garantidas para que o caso de uso funcione com sucesso.
  - É finalizado com uma pós-condição que representa os resultados observados e o estado final do sistema após o término do caso de uso.
  - Descreve o fluxo de processo de um sistema baseado nas suas possibilidades de utilização.
- Os casos de testes derivados de casos de uso são muito úteis **na descoberta de defeitos no fluxo do processo** durante a utilização do sistema no mundo real.

- Casos de uso muitas vezes são tratados como cenários e são **úteis para construir testes de aceite** com a participação do usuário final.
- Podem ajudar a descobrir defeitos de integração causados pela interação e interferência de diferentes componentes, que testes individuais de componentes podem não ter detectado.





# Critérios de teste funcional

---

- | Os critérios mais conhecidos da técnica de teste funcional são:
  - | Partição de Equivalência
  - | Análise do Valor Limite
  - | Sistemático
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Error Guessing
  - | Grafo Causa-Efeito



# Error Guessing

---

- i Essas técnica corresponde a uma abordagem na qual é apoiada na experiência do testadores.
- i A ideia do critério é enumerar possíveis erros ou situações propensas a erros e então definir casos de teste para explorá-los.
  - | Exemplos:
    - i Divisão por zero;
    - i Pressionar o botão “Enviar” de um formulário sem dados preenchidos;
    - i Ponteiros nulos;
    - i Parâmetros inválidos em funções ou procedimentos.
- i Não existem regras explícitas para esse critério.
- i Os testes podem ser projetados dependendo da situação.



# Critérios de teste funcional

---

- | Os critérios mais conhecidos da técnica de teste funcional são:
  - | Partição de Equivalência
  - | Análise do Valor Limite
  - | Sistemático
  - | Tabela de Decisão
  - | Teste de Transição de Estados
  - | Teste de Caso de Uso
  - | Error Guessing
  - | Grafo Causa-Efeito

# Grafo causa-efeito

---

A teoria dos grafos estuda objetos combinatórios, pois os mesmos são bons modelos para representar vários problemas em vários ramos da matemática, da informática, da engenharia, da química, da psicologia e da indústria.

- i Uma das limitações dos critérios de testes anteriores é que eles não exploram combinações de dados de entrada.
- i Mas... testar combinações de entrada de dados NÃO É TRIVIAL!
  - O número de combinações pode ser muito grande...
- i O critério Grafo Causa-Efeito ajuda na definição de um conjunto de casos de teste que exploram **ambiguidades** e **incompletude** nas especificações.
- i Grafo: é uma linguagem gráfica formal na qual a especificação de requisitos de uma ou mais funcionalidades do sistema é traduzida com a finalidade de **derivar casos de teste**.

# Grafo causa-efeito

---

## Processo de criação do grafo:

- 1) Dividir a especificação do *software* em partes, pois a construção do grafo para grandes especificações torna-se bastante complexa.
- 2) Identificar as **causas** e **efeitos** na especificação.
  - As causas correspondem às entradas, estímulos, ou qualquer coisa que provoque uma resposta do sistema em teste;
  - Os efeitos correspondem às saídas, mudanças no estado do sistema ou qualquer resposta observável.
- 3) Analisar a semântica da especificação e transformar em um grafo booleano – o Grafo Causa-Efeito – que liga as causas e os efeitos.
- 4) Adicionar anotações ao grafo, as quais descrevem combinações das causas e efeitos que são impossíveis devido a restrições sintáticas ou do ambiente.
- 5) Converter o grafo em uma tabela de decisão, na qual cada coluna representa um caso de teste.
- 6) Converter as colunas da tabela de decisão em casos de teste.



# Grafo Causa-Efeito

---

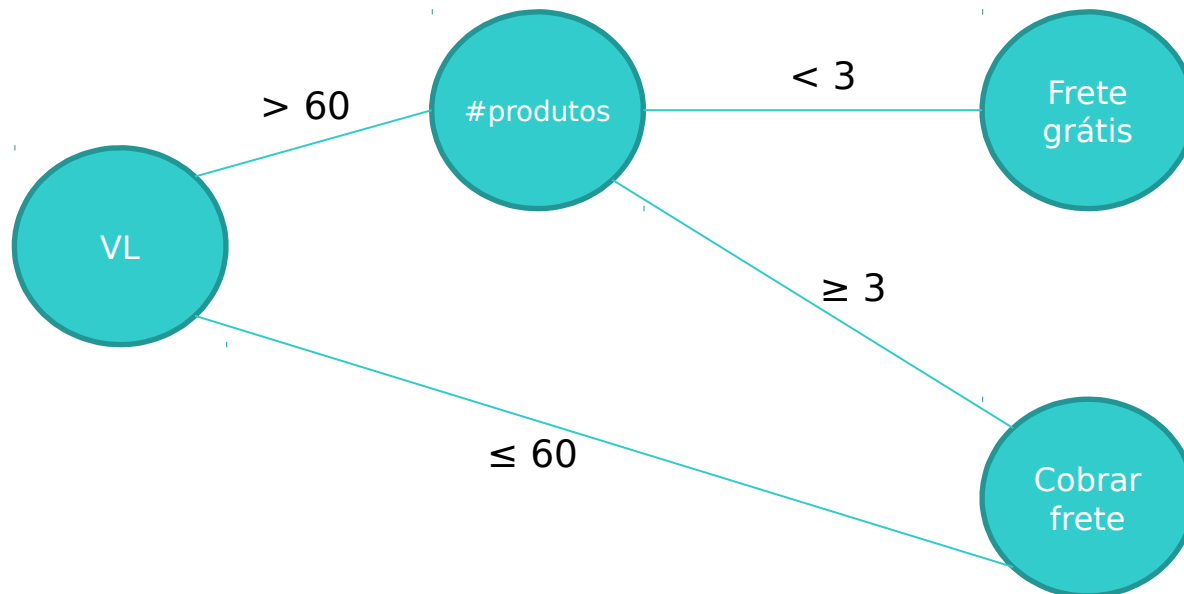
- i Para exemplificar seu uso, consideraremos a seguinte descrição:

“Em um programa de compras pela Internet, a cobrança ou não do frete é definida seguindo tal regra: se o valor da comprar for maior que R\$ 60,00 e foram comprados menos que 3 produtos, o frete é gratuito. Caso contrário, o frete deverá ser cobrado.”

# Grafo Causa-Efeito

## Exemplo1

1. Causas (condições de entrada) e efeitos (ações realizadas às diferentes condições de entrada) são relacionados, atribuindo-se um identificador para cada um.  
Causa: VL (valor da compra) > 60 **E** #produtos < 3
2. Em seguida, um grafo de causa efeito (árvore de decisão) é desenhado.





# Grafo Causa-Efeito

## Exemplo1

---

3. Neste ponto, transforma-se o grafo em uma tabela de decisão conforme a seguir:

Causa	Valor da compra	> 60	> 60	≤ 60
	#Produtos	< 3	≥ 3	-
Efeito	Cobrar frete		V	V
	Frete grátis	V		

4. As regras da tabela de decisão são então convertidas em casos de teste. Para elaboração dos casos de teste, devemos seguir todas as regras extraídas da tabela. Esse critério deve ser combinado com os dois outros já apresentados (Particionamento em classes de equivalência e AVL).

# Grafo Causa-Efeito

## Exemplo1

---

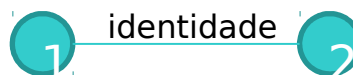
- i Os casos de teste definidos a seguir refletem somente as regras extraídas da tabela de decisão.
- i Casos de Teste (valor, #produtos, resultado\_esperado)
  - | (61, 2, “frete grátis”);
  - | (61, 3, “cobrar frete”);
  - | (60, qualquer quantidade, “cobrar frete”)

# Grafo causa-efeito

---

- i Considerando que cada nó do grafo pode assumir os valores 0 e 1, que representam, respectivamente, ausência ou presença do estado, a notação utilizada no Grafo Causa-Efeito é composta dos operadores apresentados na figura e descritos a seguir:

Função **identidade**: se o nó “1” é 1, então o nó “2” é 1; senão o nó “2” é 0.

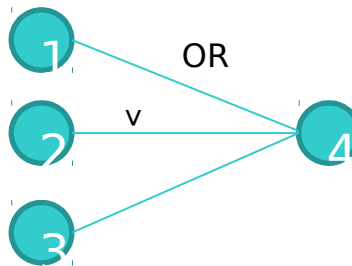


Função **not**: se o nó “1” é 1, então o nó “2” é 0; senão o nó “2” é 1.

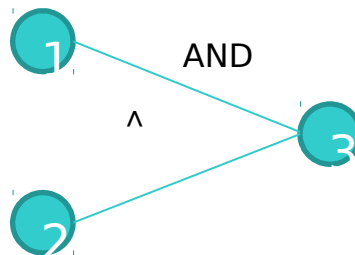


# Grafo causa-efeito

Função **or**: se o nó “1” ou “2” ou “3” é 1, então o nó “4” é 1; senão o nó “4” é 0.



Função **and**: se ambos os nós “1” e “2” são 1, então o nó “3” é 1; senão o nó “3” é 0.



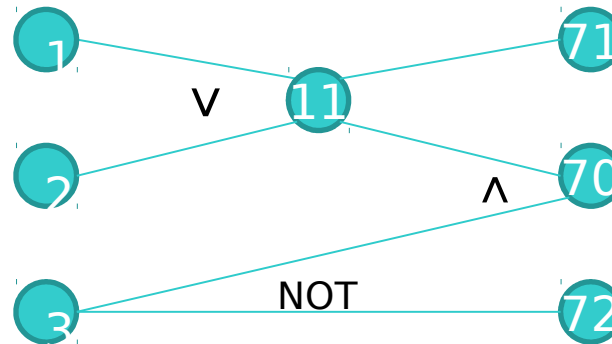
# Grafo causa-efeito

---

- i Para ilustrar a combinação desses operadores em um grafo, considere a especificação do programa “Imprime Mensagens”:
  - | O programa lê dois caracteres e, de acordo com eles, mensagens serão impressas da seguinte forma: o primeiro caractere deve ser um “A” ou um “B”. O segundo caractere deve ser um dígito. Nessa situação, o arquivo deve ser atualizado. Se o primeiro caractere é incorreto, enviar a mensagem X. Se o segundo caractere é incorreto, enviar a mensagem Y.
  - | As causas são:
    1. Caractere na coluna 1 é “A”;
    2. Caractere na coluna 1 é “B”;
    3. Caractere na coluna 2 é um dígito.
  - | Os efeitos são:
    70. A atualização é realizada;
    71. A mensagem X é enviada;
    72. A mensagem Y é enviada.

# Grafo causa-efeito

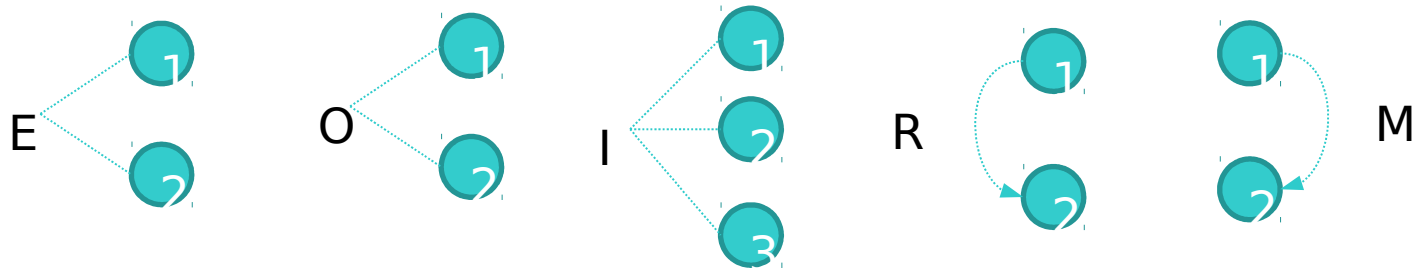
- i O Grafo Causa-Efeito correspondente é apresentado na figura a seguir.



- i Para verificar se ele corresponde à especificação basta atribuir os possíveis valores 0 e 1 às causas e observar se os efeitos assumem os valores corretos.
- i Apesar de o grafo da figura corresponder ao exemplo, ele contém uma combinação de causas que é impossível de ocorrer, uma vez que “1” e “2” não podem possuir o valor 1 simultaneamente.
- i Esta e outras situações ocorrem comumente na prática.

# Grafo causa-efeito

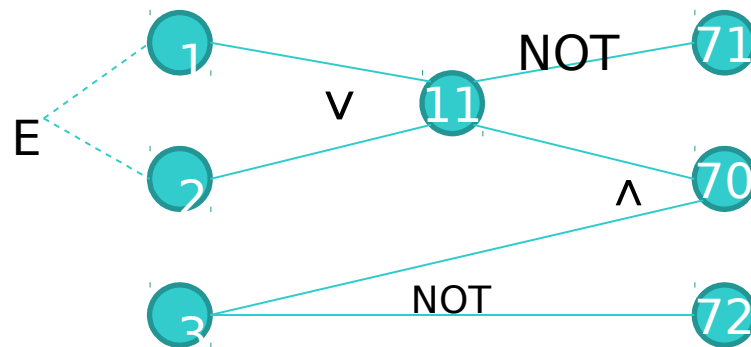
- i Para contornar esses problemas, a notação da figura abaixo pode ser utilizada, sendo que elas representam o seguinte:



- Restrição E: no máximo um entre “1” e “2” pode ser igual a 1 (ou seja, “1” e “2” não podem ser 1 simultaneamente).
- Restrição O: um e somente um entre “1” e “2” deve ser igual a 1.
- Restrição I: no mínimo um entre “1”, “2” e “3” deve ser igual a 1 (ou seja, “1”, “2” e “3” não podem ser 0 simultaneamente).
- Restrição R: para que “1” seja igual a 1, “2” deve ser igual a 1 (ou seja, é impossível que “1” seja 1 se “2” for 0).
- Restrição M: se o efeito “1” é 1 o efeito “2” é forçado a ser 0.

# Grafo causa-efeito

- i Assim, considerando a representação dessas restrições e considerando também ser impossível que as causas “1” e “2” estejam presentes simultaneamente, mas que é possível ambas não estarem presentes, o grafo é refeito para denotar essa situação e é apresentado





# Grafo causa-efeito

---

- i O próximo passo é estudar sistematicamente o grafo e construir uma tabela de decisão.
- i Essa tabela mostra os efeitos (ações) que ocorrem para todas as possíveis combinações de causas (condições).
- i Portanto, se podem ocorrer  $n$  causas, a tabela de decisão contém  $2n$  entradas.
- i O seguinte procedimento pode ser utilizado para elaborar a tabela de decisão:
  1. Selecionar um efeito para estar no estado presente, isto é, com valor 1.
  2. Rastrear o grafo para trás, encontrando todas as combinações de causas (sujeitas a restrições) que fazem com que esse efeito seja 1.
  3. Criar uma coluna na tabela de decisão para cada combinação de causa.
  4. Determinar, para cada combinação, os estados de todos os outros efeitos, anotando na tabela.

# Grafo causa-efeito

---

- i Ao executar o passo 2, fazer as seguintes considerações:
  - | Quando o nó for do tipo OR e a saída deva ser 1, nunca atribuir mais de uma entrada com valor 1 simultaneamente. O objetivo disso é evitar que alguns erros não sejam detectados pelo fato de uma causa mascarar outra.
  - | Quando o nó for do tipo AND e a saída deva ser 0, todas as combinações de entrada que levem à saída 0 devem ser enumeradas. No entanto, se a situação é tal que uma entrada é 0 e uma ou mais das outras entradas é 1, não é necessário enumerar todas as condições em que as outras entradas sejam iguais a 1.
  - | Quando o nó for do tipo AND e a saída deva ser 0, somente uma condição em que todas as entradas sejam 0 precisa ser enumerada, (Se esse AND estiver no meio do grafo, de forma que suas entradas estejam vindo de outros nós intermediários, pode ocorrer um número excessivamente grande de situações nas quais todas as entradas sejam 0).

# Grafo causa-efeito

---

Seguindo essas diretrizes, a tabela de decisão para o grafo é apresentada a seguir:

1	0	0	1	0	1
2	0	0	0	1	0
3	0	1	1	1	0
70	0	0	1	1	0
71	1	1	0	0	0
72	1	0	0	0	1

O passo final é converter cada coluna da tabela de decisão em um **caso de teste**, como será visto no exemplo a seguir.

# Grafo causa-efeito

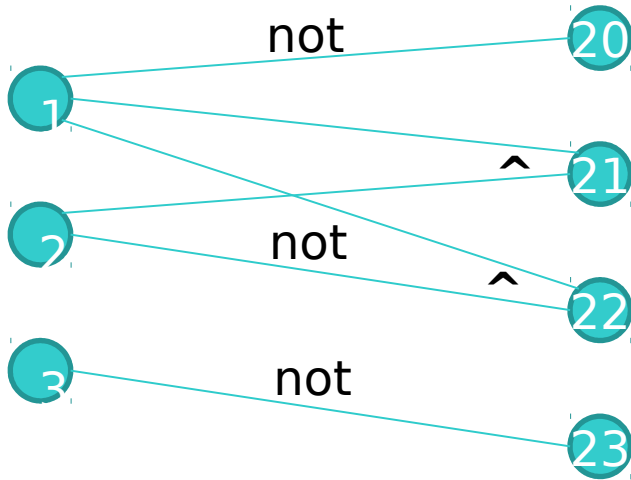
## Exemplo de aplicação

---

- i Considerando a especificação do programa “Cadeia de Caracteres”, podem ser identificadas as seguintes causas:
  - 1. Inteiro positivo no intervalo [1..20]
  - 2. Caractere a ser procurado na cadeia
  - 3. Procurar outro caractere
- i E os seguintes efeitos:
  - 20. Inteiro fora do intervalo
  - 21. Posição do caractere na cadeia
  - 22. Caractere não encontrado
  - 23. Término do programa
- i O Grafo Causa-Efeito desse exemplo é apresentado a seguir juntamente com a respectiva tabela de decisão.

# Grafo causa-efeito

## Exemplo de aplicação



Grafo Causa-Efeito da especificação Cadeia de Caracteres

1	0	1	1	-
2	-	1	0	-
3	-	1	1	0
20	1	0	0	0
21	0	1	0	0
22	0	0	1	0
23	0	0	0	1

Tabela de decisão referente ao grafo Cadeia de Caracteres

# Grafo Causa-Efeito

## Exemplo2

Sistema Sis	
Usuário	<input type="text"/>
Senha	<input type="text"/>
Digite os caracteres	<div><div>Chars</div><input type="text"/></div>
<input type="button" value="Login"/>	<input type="button" value="Cancelar"/> <input type="button" value="Mudar senha"/> <input type="button" value="Esqueci senha"/>

- Causas

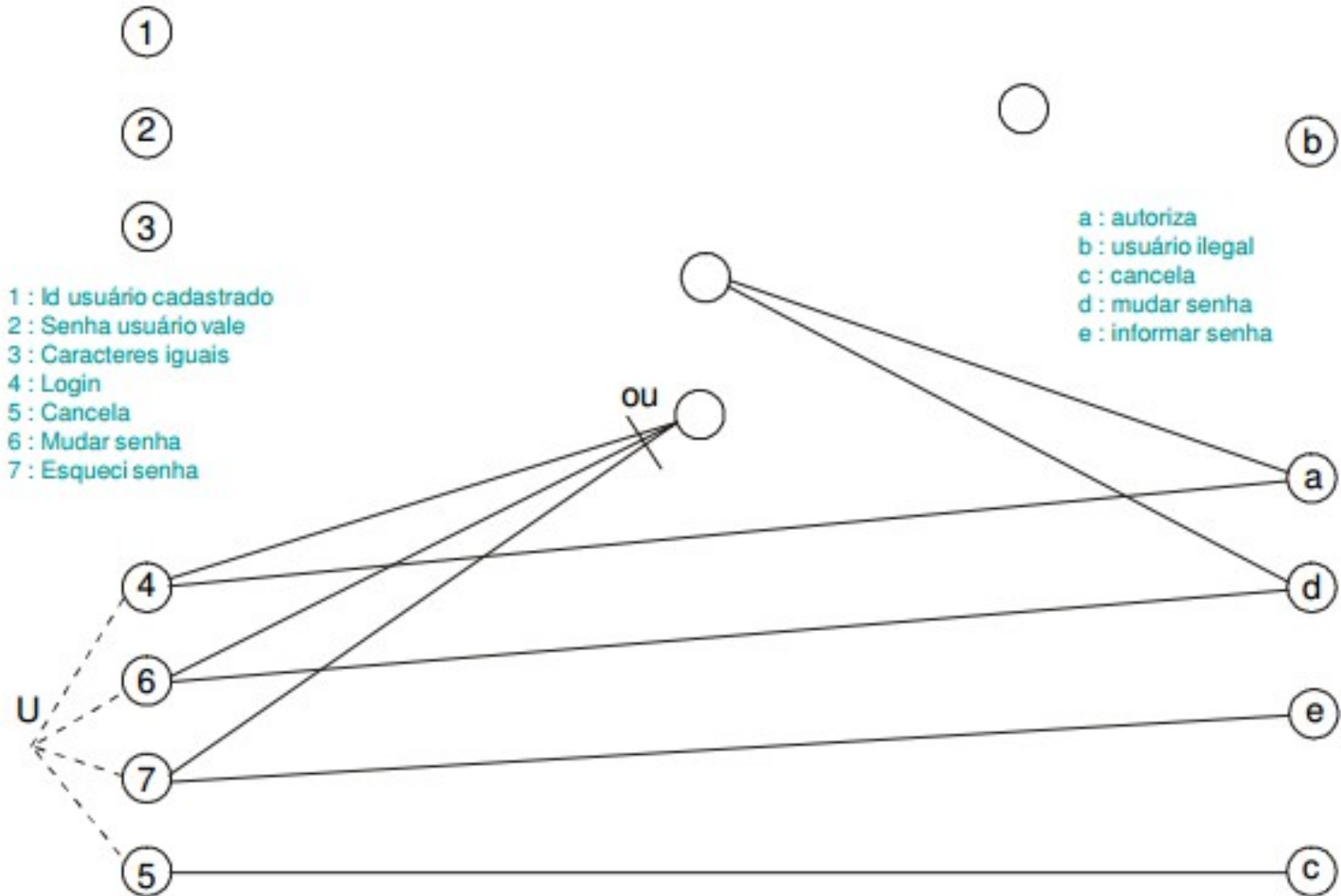
- 1 : Nome usuário cadastrado
- 2 : Senha usuário vale
- 3 : Caracteres iguais
- 4 : Login
- 5 : Cancela
- 6 : Mudar senha
- 7 : Esqueci senha

- Efeitos

- a : autoriza
- b : usuário ilegal
- c : cancela
- d : mudar senha
- e : informar senha

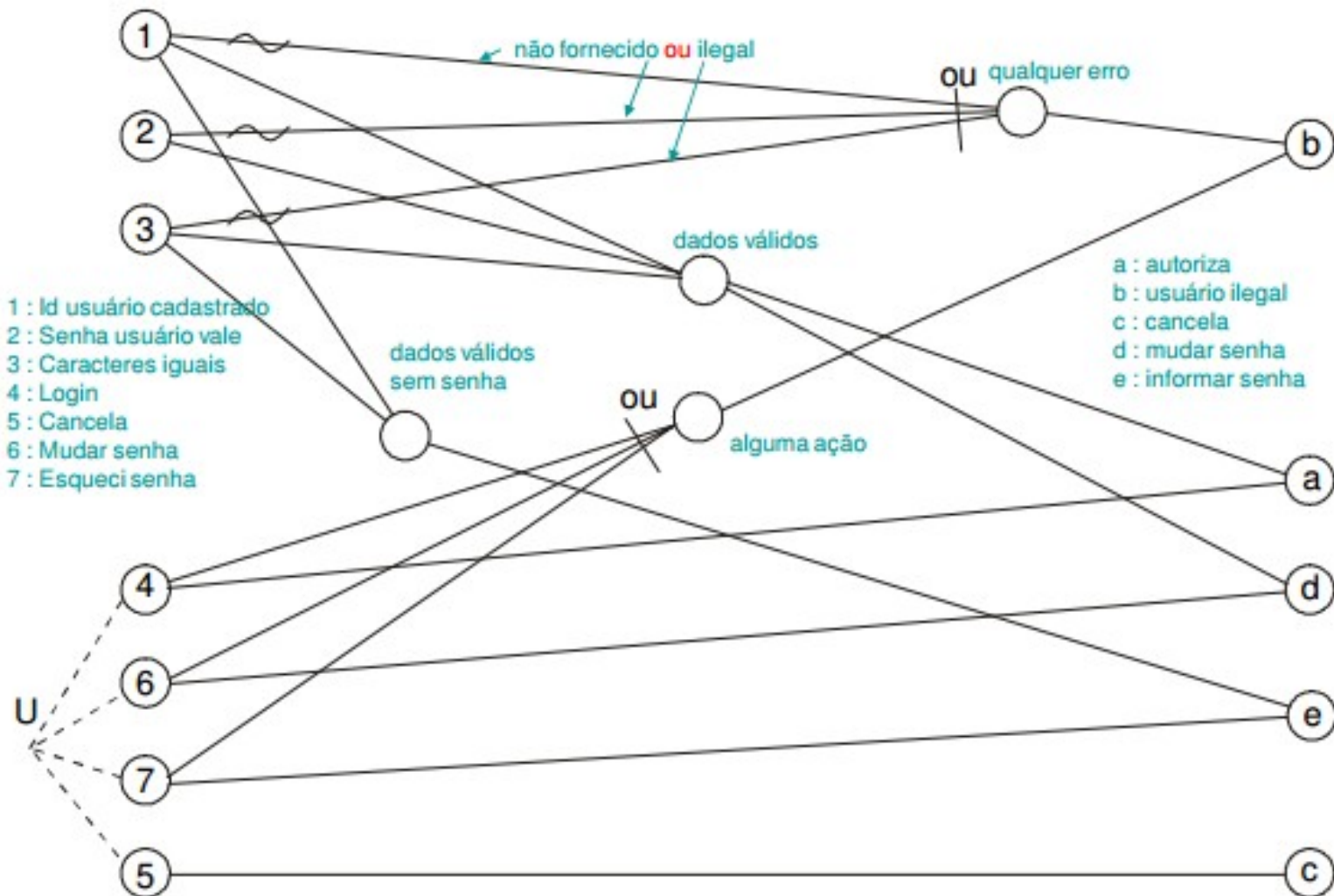
# Grafo Causa-Efeito

## Exemplo2



# Grafo Causa-Efeito

## Exemplo2





# Grafo Causa-Efeito

## Exemplo2

	1	2	3	4	5	6	7	8	9	10	11	12	13
Usuário	-	-	-	-	s	s	s	s	s	n	n	n	-
Senha	-	-	-	-	s	s	-	n	n	-	-	-	-
Letras	-	n	n	n	s	s	s	s	s	s	s	s	-
Cancela	s	n	n	n	n	n	n	n	n	n	n	n	n
Login	-	s	n	n	s	n	n	s	n	s	n	n	n
Muda	-	-	s	n	-	s	n	-	s	-	s	n	n
Esqueci	-	-	-	s	-	-	s	-	-	-	-	s	n
Autoriza					x								
Illegal		x	x	x				x	x	x	x	x	
Cancela	x												
Muda						x							
Informa							x						
Impossível													x
	64	16	8	4	4	2	2	4	2	8	4	2	8

=128



Porque esta coluna?




# Grafo causa-efeito

## Avaliação do critério

---

- i A vantagem do Grafo Causa-Efeito é que esse critério exercita combinações de dados de teste que, possivelmente, não seriam considerados.
- i Além disso, os resultados esperados são produzidos como parte do processo de criação do teste, ou seja, eles fazem parte da própria tabela de decisão.
- i As dificuldades do critério estão na complexidade em se desenvolver o grafo booleano, caso o número de causas e efeitos seja muito grande, e também na conversão do grafo na tabela de decisão, embora esse processo seja algorítmico e existam ferramentas de auxílio.
- i Uma possível solução para isso é tentar identificar subproblemas e desenvolver subgrafos Causa-Efeito para eles.



# Grafo causa-efeito

## Avaliação do critério

---

- i A eficiência desse critério, assim como os outros critérios funcionais, depende da **qualidade da especificação**, para que sejam identificadas as causas e os efeitos e, também, da habilidade e experiência do testador.
- i Uma especificação muito detalhada pode levar a um grande número de causas e efeitos e, por outro lado, uma especificação muito abstrata pode não gerar dados de teste significativos.

# Referências

---

- i Justo, Daniela Sbizzera Profª Dra. Introdução a teste de software. IFSC. Gaspar.
- i Anne Caroline O. Rocha - Tester Certified - BSTQB - NTI|UFPB
- i Delamaro, M.E.; Maldonado, J.C.; Jino, M. **Introdução ao Teste de Software**. São Paulo: Elsevier, 2007.
- i Coursera. **Introdução ao Teste de Software: Particionamento em Classes de Equivalência**. Acesso em: 14 set. 2018. Disponível em: <  
<https://www.coursera.org/learn/intro-teste-de-software/lecture/BCuDd/particionamento-em-classes-de-equivalencia>
- i Coursera. **Introdução ao Teste de Software: Análise do Valor Limite**. Acesso em: 14 set. 2018. Disponível em:  
<https://www.coursera.org/lecture/intro-teste-de-software/analise-do-valor-limite-oSV2x>