

Mergulhando na CTFL!

Análise de Valor de Limite (Boundary Value Analysis - BVA) [↗](#)

✅ **Conceito simples:** Essa técnica testa **os limites dos valores válidos e inválidos** de uma entrada. Exemplo clássico: idade de cadastro entre 18 e 60 anos.

✅ **Conceito mais técnico:** [↗](#)

A Análise de Valor de Limite é uma técnica **baseada em caixa preta**, que **assume que os erros se concentram nos limites das partições válidas e inválidas**. Essa técnica é normalmente aplicada **junto com Particionamento de Equivalência** (mas você não precisa explicar essa outra técnica, só mencionar, se quiser parecer mais completo).

Por que acontece erro nos limites? [↗](#)

- **Erros de digitação de código**, como usar `>` em vez de `>=`.
- **Lógicas mal implementadas** de validação de entrada.
- **Arredondamentos**, truncamentos ou limitações internas.

Por exemplo:

```
1 python

if idade > 18:      # A lógica deveria permitir 18, mas não permite.
```

Tipos de valores testados [↗](#)

Para uma **faixa de valores de 1 a 100**, você testa:

- **Limite inferior - 1:** 0 ❌ (inválido)
- **Limite inferior:** 1 ✅
- **Limite inferior + 1:** 2 ✅
- **Limite superior - 1:** 99 ✅
- **Limite superior:** 100 ✅
- **Limite superior + 1:** 101 ❌ (inválido)

Formato de tabela sugerido: [↗](#)

Valor testado	Motivo	Esperado
0	Abaixo do limite inferior	Rejeitado ❌
1	No limite inferior	Aceito ✅
2	Acima do limite inferior	Aceito ✅
99	Abaixo do limite superior	Aceito ✅
100	No limite superior	Aceito ✅
101	Acima do limite superior	Rejeitado ❌

⚙️ Onde essa técnica é mais útil? [↗](#)

- Validação de **formulários com faixa de idade, preço, nota, etc.**
- Sistemas **bancários** (limite mínimo/máximo de saque).
- **Aplicações críticas** onde **um erro nos extremos pode causar prejuízo.**

! Variações avançadas (caso queiram te perguntar): [↗](#)

- **Limites múltiplos:** quando o campo tem mais de um limite (ex: de 1 a 100, e só números pares).
- **Limites não numéricos:** você pode aplicar em **strings** (ex: mínimo de 3 caracteres, máximo de 10).

🧠 Dica extra: quando você quiser impressionar [↗](#)

Diga algo assim na apresentação:

"Essa técnica é rápida de aplicar e aumenta a chance de encontrar defeitos críticos com poucos testes, focando onde os sistemas costumam falhar: nas bordas."

Teste de Tabela de Decisão [↗](#)

✅ Conceito mais técnico: [↗](#)

O **Teste de Tabela de Decisão** é uma técnica de **teste baseado em lógica de decisão**, usada para validar **todas as combinações possíveis de condições (entradas)** e verificar **se as ações (saídas) esperadas são realizadas** corretamente.

Ele é perfeito para situações onde o comportamento do sistema **depende de múltiplas regras combinadas.**

🧠 Por que isso é importante? [↗](#)

- Sistemas reais costumam ter **regras interdependentes.**
- O ser humano pode **esquecer combinações possíveis** na hora de escrever testes manuais.
- Garante que **nenhuma decisão lógica fique de fora** (ótimo pra detectar falhas escondidas).

📋 Estrutura de uma tabela de decisão: [↗](#)

Ela é composta por duas partes:

1. **Condições** (entradas)
2. **Ações** (saídas)

E as **colunas** representam **regras**, ou seja, **combinações possíveis** entre essas condições.

📁 Exemplo aprofundado (Sistema de Desconto) [↗](#)

Regras de negócio:

- Cliente VIP tem desconto.
- Compras acima de R\$500 têm desconto maior.
- Se não for VIP, mas comprar acima de R\$500, tem um pequeno desconto.
- Caso contrário, não tem desconto.

Regra	Cliente VIP	Compra > R\$500	Ação
R1	Não	Não	Sem desconto
R2	Não	Sim	10%
R3	Sim	Não	10%
R4	Sim	Sim	20%

✓ Como montar uma tabela de decisão: [🔗](#)

1. **Liste todas as condições** envolvidas no processo.
 2. **Enumere todas as combinações possíveis** entre essas condições (isso dá 2ⁿ, se forem só V/F).
 3. **Associe cada combinação a uma ação esperada.**
 4. **Verifique se todas as regras de negócio estão corretamente refletidas.**
-

🧠 Curiosidade técnica: [🔗](#)

Essa técnica é **essencial em sistemas com lógica complexa**, como:

- Sistemas bancários (ex: liberação de crédito)
 - Planos de saúde (ex: aprovação de exames)
 - E-commerce (ex: cálculo de frete, cupom, desconto...)
-

⚠ Erros comuns que o Teste de Tabela de Decisão ajuda a evitar: [🔗](#)

- **Conflito entre regras**
 - **Esquecer uma regra importante**
 - **Ação errada para uma condição específica**
-

🎯 Quando usar? [🔗](#)

- Quando há **muitas condições com combinações possíveis**.
 - Quando o cliente ou a equipe diz:
“Depende de várias coisas...”
-

Teste de Transição de Estado (State Transition Testing) [🔗](#)

✓ Conceito técnico: [🔗](#)

É uma técnica de **teste baseado em comportamento** que verifica se o sistema **responde corretamente a eventos** quando muda de **um estado para outro**.

Ela se baseia em diagramas de **máquinas de estados**, onde o sistema:

- Está em um **estado atual**
- Recebe um **evento**
- E **transita** para um **novo estado**

🧠 Por que isso é importante? [🔗](#)

- Muitos sistemas **não são lineares** — o comportamento muda **dependendo do estado atual**.
- Essa técnica ajuda a encontrar **erros de fluxo**, **transições não permitidas** e **respostas erradas a eventos**.
- É essencial para sistemas que **controlam processos ou respondem a ações do usuário**.

ATM Exemplo prático (Caixa eletrônico / Login de usuário): [🔗](#)

Imagine um sistema de login com os seguintes estados:

- Deslogado
- Logando
- Logado
- Bloqueado

Estado Atual	Evento	Novo Estado
Deslogado	Digita senha	Logado
Logado	Clica sair	Deslogado
Deslogado	3 tentativas erradas	Bloqueado

✂ Como representar isso (Visual): [🔗](#)

Você pode desenhar um **diagrama simples** assim:

```
[Deslogado] --(senha correta)--> [Logado]
```

```
[Deslogado] --(3 erros)-----> [Bloqueado]
```

```
[Logado] ----(sair)-----> [Deslogado]
```

✓ Esse desenho pode ser feito no PowerPoint, Canva ou até desenhado à mão com setinhas.

🔧 Onde esse tipo de teste é útil? [🔗](#)

- **Sistemas bancários:** ATM, app de saldo, login/logout
- **Aplicativos web:** Troca de status (rascunho → enviado → aprovado → arquivado)
- **Softwares embarcados:** Controle de elevadores, semáforos, sensores

- **E-commerces:** Carrinho vazio → item adicionado → pagamento iniciado → pedido finalizado
-

🔥 Falhas que ele ajuda a detectar: [🔗](#)

- Transições **ilegais** (ex: logar direto sem digitar senha)
 - Estados **não alcançáveis**
 - Eventos que **não geram transição**
 - Comportamentos errados se o sistema estiver no estado errado
-

💡 Dica de ouro: [🔗](#)

Se o sistema muda de comportamento com base em algo que **já aconteceu**, use teste de transição de estado.

Exemplo: “Se o usuário já comprou antes, ele vê outra tela.” — isso é estado.
