Strecht Simulados · Guia de preparação



Guia de preparação ISTQB CTFL

Tudo o que a prova cobra, organizado por domínio: conceitos-chave, comparações que confundem, pegadinhas reais, gatilhos do enunciado, questões comentadas e um plano para chegar pronto ao exame.

6
capítulos por domínio

36
questões comentadas

65%
nota de corte

6
provas completas no curso

ANTES DE COMEÇAR

Como usar este guia

Este guia foi feito para ser estudado junto com as 6 provas completas do curso ISTQB CTFL: o guia explica o que cai e onde estão as armadilhas; as provas treinam o ritmo e mostram o que você ainda não domina.

Quando	O que fazer
Semana 1	Faça o teste diagnóstico (página seguinte) e leia os capítulos 1 e 2. No fim da semana, faça a Prova 1 do curso no tempo do exame.
Semana 2	Leia os demais capítulos, começando pelos domínios em que você errou mais no diagnóstico. Faça as provas 2 e 3.
Semana 3	Releia os alertas de pegadinha e as tabelas de comparação. Faça as provas 4 e 5 e revise cada erro pela explicação.
Semana 4	Estude a folha de cola, faça a Prova 6 e refaça as provas abaixo de 80%. Agende o exame quando estiver acima de 80% em todas (a nota de corte é 65%).

Mapa do exame

Cada prova do curso tem 40 questões e 60 minutos, no formato do exame. Referência de aprovação: 65%.

Domínio (capítulo)	Peso	Neste guia
1. Análise e Modelagem de Teste	28%	18 conceitos · 14 pegadinhas
2. Ferramentas de Teste	5%	12 conceitos · 10 pegadinhas
3. Fundamentos de Teste	20%	18 conceitos · 14 pegadinhas
4. Gerenciamento das Atividades de Teste	22%	18 conceitos · 14 pegadinhas
5. Teste Estático	10%	18 conceitos · 14 pegadinhas
6. Teste ao Longo do Ciclo de Vida de Desenvolvimento	15%	18 conceitos · 14 pegadinhas

Como ler os capítulos: os **tópicos-chave** são o que mais cai; a **tabela de memorização** separa o que a prova costuma confundir; os **alertas de prova** mostram as armadilhas das alternativas erradas; os **gatilhos** ligam palavras do enunciado à resposta; e o **checklist** diz se você está pronto para o domínio.

Estude um pouco todo dia, de graça

Instagram @strecht.simulados

Um post de prova por dia: questão comentada, pegadinhas e o “não confunda” de cada certificação.

Seguir no Instagram →

YouTube Strecht Simulados

Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.

Inscrever-se no canal →

DIAGNÓSTICO

Você já sabe isso?

Responda às 12 questões sem consultar nada. As respostas, com a explicação e o capítulo para revisar, estão no fim do guia. Comece o estudo pelos capítulos em que errou.

D1 · Análise e Modelagem de Teste

Um componente tem 10 instruções executáveis, e os testes executaram 8 delas. Qual é a cobertura de instrução?

A. 8%
B. 80%
C. 20%
D. 100%

D2 · Análise e Modelagem de Teste

Uma tabela de decisão tem três condições booleanas independentes. Quantas regras tem a tabela completa, antes de qualquer simplificação?

A. 9 regras
B. 3 regras
C. 6 regras
D. 8 regras

D3 · Análise e Modelagem de Teste

Foram identificados 8 valores limite para um campo, e os testes exercitaram 6 deles. Qual é a cobertura de BVA?

A. Cobertura de 25%
B. Cobertura de 75%
C. Cobertura de 60%
D. Cobertura de 80%

D4 · Ferramentas de Teste

Qual categoria de ferramenta apoia o controle de requisitos, a rastreabilidade, os testes e os relatórios de progresso?

A. Ferramentas de gerenciamento
B. Ferramentas de DevOps
C. Ferramentas de desempenho
D. Ferramentas de teste estático

D5 · Ferramentas de Teste

Qual categoria de ferramenta apoia o processo de revisão e a análise estática?

A. Ferramentas de colaboração
B. Ferramentas de teste estático
C. Ferramentas DevOps
D. Ferramentas de teste não funcional

D6 · Ferramentas de Teste

Qual categoria de ferramenta apoia o pipeline de entrega, a construção automatizada e a integração contínua?

A. Ferramentas de teste estático
B. Ferramentas de DevOps
C. Ferramentas de colaboração
D. Ferramentas de modelagem

D7 · Fundamentos de Teste

Qual afirmação diferencia corretamente garantia da qualidade (QA) e teste?

A. QA foca no processo; teste avalia o produto
B. QA e teste são exatamente a mesma atividade
C. QA só acontece após a entrega do produto
D. Teste foca no processo; QA avalia o produto

D8 · Fundamentos de Teste

Durante a análise de teste, a equipe encontra ambiguidades e contradições nos requisitos. O que isso mostra sobre essa atividade?

A. Ela deve ser adiada até o código existir
B. Ela só produz os casos de teste finais
C. Ela também avalia a base de teste
D. Ela substitui as revisões formais

D9 · Fundamentos de Teste

Qual atividade localiza, analisa e remove a causa de uma falha no código?

A. Teste de confirmação
B. Teste de regressão
C. Depuração do código
D. Análise de teste

D10 · Gerenciamento das Atividades de Teste

Qual destas é uma métrica de defeitos?

A. Número de casos planejados
B. Densidade de defeitos por módulo
C. Número de reuniões da equipe
D. Custo de licença das ferramentas

D11 · Gerenciamento das Atividades de Teste

Uma equipe estimou a = 6, m = 9 e b = 18 horas para testar um relatório. Usando (b − a) / 6, qual é o desvio-padrão da estimativa?

A. 3 horas
B. 10 horas
C. 2 horas
D. 12 horas

D12 · Gerenciamento das Atividades de Teste

Qual atividade ocorre na conclusão do teste?

A. Escolher as técnicas de modelagem
B. Preparar os dados de teste da execução
C. Definir as condições de teste iniciais
D. Arquivar o testware e as lições

01

CAPÍTULO 1 · 28% DA PROVA

Análise e Modelagem de Teste

Este é o capítulo mais pesado: 28% da prova, cerca de 11 questões, e o único com muita conta. Preciso dominar particionamento de equivalência, análise de valor limite de 2 e 3 valores, tabela de decisão, transição de estado, cobertura de instrução e de ramificação, técnicas baseadas em experiência e as abordagens colaborativas (histórias de usuário, critérios de aceite, ATDD). As questões trazem uma faixa de valores, uma regra de negócio ou um trecho de lógica e pedem valores de teste, número mínimo de testes ou percentual de cobertura. Onde mais erro: vizinho interno x externo na BVA, o 100% de instrução que não é 100% de ramificação, e marcar o percentual NÃO coberto no lugar do coberto.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 8 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Particionamento de equivalência (EP)

Uma partição reúne valores que o objeto de teste processa da mesma forma. Partições não se sobrepõem e não podem ser vazias; podem ser contínuas ou discretas, ordenadas ou não, e de textos, listas ou tipos. Faixas com o mesmo resultado, mas separadas (meia até 17 e meia a partir de 60), são partições diferentes; um intervalo fechado tem uma partição válida e duas inválidas.

Cobertura e mínimo de testes em EP

Cobertura = partições exercitadas / partições identificadas. Com um único campo, o mínimo é o total de partições (um teste por partição). Com vários campos, um caso pode cobrir uma partição de cada campo, então o mínimo é o maior número de partições entre os campos. Partições inválidas não se combinam no mesmo caso, porque uma falha pode mascarar a outra.

Análise de valor limite (BVA)

Só se aplica a partições ordenadas. Valores limite são os extremos das partições; vizinhos ficam imediatamente antes ou depois, conforme a menor variação do dado (com uma casa decimal, o passo é 0,1). Partição aberta não tem limite de um dos lados. Cobertura = valores limite exercitados / identificados.

BVA de 2 valores

Para cada limite, testa-se o limite e o vizinho da partição adjacente (o de fora). Intervalo 1 a 100: 0, 1, 100 e 101, total de quatro valores. Idade mínima 18 sem máximo: 17 e 18.

BVA de 3 valores

Para cada limite, testa-se o limite e os dois vizinhos, um de cada lado. Intervalo 10 a 20: 9, 10, 11, 19, 20 e 21, total de seis valores. Encontra mais defeitos que a de 2 valores, como erros de igualdade.

Tabela de decisão

Combina condições e ações; cada coluna é uma regra. Nas condições, V/F (ou S/N) indicam se a condição é satisfeita e o traço (–) indica valor irrelevante; um X na linha da ação indica que a ação ocorre naquela regra. Regra com E só dispara com todas as condições verdadeiras; regra com OU só é falsa quando todas são falsas.

Tamanho, redução e cobertura da tabela

Com n condições booleanas, a tabela completa tem 2^n regras; com condições não booleanas, multiplica-se o número de valores de cada uma. Reduz-se unindo regras em que uma condição não afeta a ação e removendo combinações impossíveis ou inviáveis. Cobertura = regras exercitadas / regras viáveis; o mínimo é um teste por regra viável.

Modelo de transição de estado

Uma transição é disparada por um evento, com possível condição de guarda e ação associada. O diagrama de transição de estados mostra só as transições válidas; a tabela de estados (linhas = estados, colunas = eventos) mostra também as inválidas. Testar um evento não previsto no estado atual, como cancelar pedido já entregue, é teste de transição inválida.

Critérios de cobertura de transição de estado

Cobertura de todos os estados (visitar cada estado; a mais fraca, calculada como estados visitados / total). Cobertura de transições válidas (0-switch: exercitar cada transição válida). Cobertura de todas as transições (inclui tentativas de transições inválidas; a mais forte, recomendada para software crítico). Estado final limita o caminho e pode exigir mais de um teste.

Cobertura de instrução

Instruções executadas / total de instruções executáveis (8 de 10 = 80%). Limitação: 100% de instrução não garante testar ramos vazios nem toda a lógica de decisão. Um IF/ELSE com uma instrução em cada ramo precisa de 2 testes.

Cobertura de ramificação

Ramos exercitados / total de ramos. Ramo é a transferência de controle entre dois nós do grafo de fluxo de controle, condicional ou incondicional; laços também têm ramos. Switch com três casos e default tem 4 ramos; duas decisões IF independentes sem ELSE pedem só 2 testes. 100% de ramificação implica 100% de instrução, mas não o contrário.

Forças e limites da caixa-branca

Ponto forte: considera toda a implementação e mede cobertura. Limite: deriva testes do código existente, então não detecta requisitos que nunca foram implementados; por isso combina-se com caixa-preta. Medir cobertura depois dos testes de caixa-preta serve para achar código não exercitado, em qualquer nível.

Suposição de erro

Baseia-se em erros, defeitos e falhas conhecidos: histórico do sistema, sistemas parecidos e falhas típicas (divisão por zero, campos vazios). Ataques de falha são a forma metódica, com listas de erros e falhas possíveis, como entradas muito longas e caracteres especiais.

Teste exploratório baseado em sessão

O testador aprende sobre o produto enquanto projeta e executa, guiado por uma carta de teste com objetivo e tempo definidos. Folhas de sessão registram carta, tempo, observações, problemas e perguntas, e cada sessão termina com debriefing. Útil com especificações escassas, prazos curtos ou para complementar técnicas formais; depende de experiência.

Teste baseado em lista de verificação

Testes criados a partir de uma lista de itens a verificar (por exemplo, de usabilidade), atualizada com a experiência. A lista não deve ser longa demais e usa itens de alto nível; evitar itens checáveis por ferramenta, itens genéricos demais e itens que funcionam melhor como critérios de entrada ou saída.

Histórias de usuário: 3 Cs e INVEST

Os 3 Cs são Cartão, Conversa e Confirmação. Boas histórias seguem INVEST: Independente, Negociável, Valiosa, Estimável, Pequena (Small) e Testável. A escrita é colaborativa entre negócio, desenvolvimento e teste, com técnicas como brainstorming e mapas mentais.

Critérios de aceite

Definem o escopo e as condições de aceite da história, ajudam a chegar a consenso, descrevem cenários positivos e negativos e servem de base para os testes. Devem estar definidos antes de a história entrar no desenvolvimento (costumam fazer parte da DoR). Formatos: orientado a cenário (Dado/Quando/Então) e orientado a regra (lista de regras a verificar).

ATDD

No desenvolvimento orientado a testes de aceite, a equipe escreve os testes a partir dos critérios de aceite antes de implementar a história, geralmente com exemplos positivos e negativos.

Macetes para memorizar

💡 BVA 2 x 3 valores

Dois é dupla: eu e o vizinho de fora. Três é sanduíche: vizinho, eu, vizinho.

💡 Contar valores BVA num intervalo fechado

2 valores → 4 testes; 3 valores → 6 testes (dois limites vezes 2 ou 3).

💡 Tamanho da tabela de decisão

Booleana: 2 elevado a n. Mista: multiplica os valores de cada condição. Nunca some.

💡 Instrução x ramificação

Ramificação manda na instrução: 100% de ramo garante 100% de instrução, nunca o contrário.

💡 Força da cobertura de estados

Estados < transições válidas (0-switch) < todas as transições.

💡 INVEST

Independente, Negociável, Valiosa, Estimável, Small (pequena), Testável; o T é sempre Testável.

💡 Técnicas de experiência

Suposição = LEMBRO de erros; Exploratório = APRENDO enquanto testo; Lista de verificação = SIGO a lista.

💡 Fórmula única de cobertura

Cobertura = o que exercitei / o que existe (partições, limites, regras, estados, instruções, ramos).

Tabela de memorização: não confunda

Quando é...	...e quando é
BVA de 2 valores Limite + vizinho de fora: faixa 1 a 100 → 0, 1, 100, 101 (quatro valores).	**BVA de 3 valores** Limite + os dois vizinhos: faixa 10 a 20 → 9, 10, 11, 19, 20, 21 (seis valores).
Cobertura de instrução Conta instruções executadas; um IF sem ELSE com teste verdadeiro já dá 100%.	**Cobertura de ramificação** Conta ramos, inclusive o ramo vazio do IF sem ELSE; o mesmo teste dá só 50%.

Cobertura de transições válidas Exercitar cada transição válida do modelo (0-switch).	**Cobertura de todas as transições** Incluir também tentativas de transições inválidas; recomendada para software crítico.
Diagrama de transição de estados Visualizar as transições válidas.	**Tabela de estados** Mostrar explicitamente também as transições inválidas (células sem transição).
Critério orientado a cenário Escrito como Dado/Quando/Então.	**Critério orientado a regra** Escrito como lista de regras a verificar.
Suposição de erro Testes a partir de erros e falhas já vistos ou de listas de ataques de falha.	**Teste exploratório** Carta de teste e sessão com tempo definido, aprendendo enquanto testa, com debriefing no fim.
Caixa-branca Avaliar a implementação e medir cobertura de código.	**Caixa-preta** Partir da especificação e revelar requisitos que não foram implementados.

Números que caem

Item	Valor
Peso do capítulo na prova	**28% (cerca de 11 questões)**
Regras da tabela completa com 3 condições booleanas	**8 (2^3)**
Valores de BVA de 2 valores num intervalo fechado	**4**
Valores de BVA de 3 valores num intervalo fechado	**6**
Ramos de um switch com três casos e default	**4**
IF sem ELSE testado só com condição verdadeira	**100% de instrução e 50% de ramificação**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** IF sem ELSE, teste com condição verdadeira: marcar 100% de cobertura de ramificação.
Como não cair: 100% é a cobertura de instrução; de ramificação são 50%, pois o ramo falso não foi exercitado.

⚠ **Alerta de prova:** Aceitar que 100% de instrução implica 100% de ramificação.
Como não cair: É o inverso: ramificação 100% leva a instrução 100%. O IF sem ELSE derruba a afirmação contrária.

⚠ **Alerta de prova:** Duas decisões IF independentes: responder 4 testes para 100% de ramificação.
Como não cair: 4 seria todas as combinações. Como são independentes, 2 testes (tudo verdadeiro, tudo falso) cobrem os ramos.

⚠ **Alerta de prova:** Switch com três casos e default: contar 3 ramos.
Como não cair: O caso padrão também é ramo: são 4.

⚠ **Alerta de prova:** BVA de 2 valores no limite inferior 18: escolher 18 e 19.
Como não cair: O vizinho é o da partição adjacente, o de fora: 17 e 18. O 19 está dentro da partição válida.

⚠ **Alerta de prova:** BVA de 3 valores no limite -10: escolher -12, -11 e -10.
Como não cair: São o limite e um vizinho de cada lado: -11, -10 e -9.

⚠ **Alerta de prova:** Nota de 0,0 a 10,0 com uma casa decimal: usar 10 e 11 no limite superior.
Como não cair: O vizinho segue a precisão do dado: 10,0 e 10,1.

⚠ **Alerta de prova:** Juntar faixas com o mesmo preço numa só partição (meia de 0 a 17 e a partir de 60).
Como não cair: Faixas não contínuas são partições separadas: são 3 partições válidas.

⚠ **Alerta de prova:** Dividir um intervalo válido único (18 a 65) em várias partições válidas ou esquecer uma das inválidas.
Como não cair: Intervalo fechado: uma válida e duas inválidas (abaixo e acima).

⚠ **Alerta de prova:** Lista de tipos com tratamentos diferentes (Corrente, Poupança, Salário): 1 válida e 1 inválida.
Como não cair: Cada valor tratado de forma diferente é uma partição válida: 3 válidas e 1 inválida.

⚠ **Alerta de prova:** Campo A com 3 partições e B com 4: responder 12 (produto) ou 7 (soma).
Como não cair: Combinando valores nos mesmos casos, o mínimo é o maior número de partições: 4.

⚠ **Alerta de prova:** Condições Novo/Regular/VIP e à vista sim/não: responder 8 (2^3) ou 5 (soma).
Como não cair: Condições não booleanas: multiplique os valores, 3 × 2 = 6 regras.

⚠ **Alerta de prova:** Marcar o percentual não coberto (20%, 25%) como cobertura.
Como não cair: Cobertura é sempre exercitados / identificados; o complemento é distrator clássico.

⚠ **Alerta de prova:** Dizer que caixa-branca detecta requisitos não implementados ou dispensa o código-fonte.
Como não cair: Ela deriva do código existente, exige acesso a ele e não vê omissões; isso é papel da caixa-preta.

Como cai na prova

📝 **Como cai:** Faixa de valores (idade, quantidade, senha, nota) e pedido de valores de teste por BVA
Como resolver: Ache os limites de cada partição; veja se é 2 ou 3 valores; aplique o passo da precisão (1 para inteiros, 0,1 para uma casa decimal); confira se a partição é aberta.

📝 **Como cai:** Número de partições ou mínimo de testes por EP
Como resolver: Conte um tratamento = uma partição (faixas não contínuas separadas, valores de lista com tratamento diferente separados); some as inválidas. Um campo: mínimo = total; vários campos: mínimo = maior número de partições.

📝 **Como cai:** Regra de negócio com E ou OU e pergunta sobre qual combinação resulta em tal ação, ou quantas regras
Como resolver: E: basta uma condição falsa para não disparar; OU: só não dispara com todas falsas. Regras: 2^n se booleanas, produto se mistas; reduza onde a condição é irrelevante.

📝 **Como cai:** Trecho de lógica (IF, IF/ELSE, switch) e pergunta de cobertura ou mínimo de testes
Como resolver: Desenhe os ramos: IF sem ELSE tem 2 ramos, switch tem um por caso mais o default. Calcule exercitados / total e lembre que instrução pode ficar em 100% com ramificação menor.

📝 **Como cai:** Modelo de estados com estado final e pergunta de mínimo de testes para cobrir transições válidas
Como resolver: Liste as transições, monte sequências partindo do estado inicial e note que, ao chegar ao estado final, o teste acaba; conte quantas sequências cobrem todas.

📝 **Como cai:** Cenário descrevendo como o testador trabalha e pedindo a técnica
Como resolver: Carta e sessão = exploratório; erros já vistos = suposição de erro; lista de itens = lista de verificação; estrutura do código = caixa-branca; partições, limites, regras ou estados = caixa-preta.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Carta de teste, sessão com tempo, debriefing	**Teste exploratório baseado em sessão**

Falhas que já viu em sistemas parecidos; divisão por zero; ataques de falha	**Suposição de erro**
Lista de itens atualizada com a experiência de projetos anteriores	**Teste baseado em lista de verificação**
Dado / Quando / Então	**Critério de aceite orientado a cenário (e BDD)**
Testes escritos a partir dos critérios antes de implementar a história	**ATDD**
Evento não previsto no estado atual	**Transição inválida; tabela de estados**
Só visitar cada estado ao menos uma vez	**Cobertura de todos os estados (a mais fraca)**
Incluir transições inválidas; software crítico	**Cobertura de todas as transições**
n condições booleanas, tabela completa	**2^n regras**
Condição que não afeta a ação	**Traço (–) e união de regras**
IF sem ELSE	**Instrução pode chegar a 100% com ramificação em 50%**
Medir cobertura de código depois dos testes de caixa-preta	**Achar código não exercitado**
Cartão, Conversa, Confirmação / Independente...Testável	**3 Cs / INVEST**
Valor mínimo definido e sem máximo	**Partição aberta: BVA só no limite que existe**

🎯 **Se você lembrar de só uma coisa deste capítulo...**

- BVA 2 valores = limite + vizinho de fora; BVA 3 valores = limite + os dois vizinhos.
- 100% de ramificação implica 100% de instrução, nunca o contrário.
- Tabela completa: 2^n para booleanas, produto dos valores para condições mistas.
- Não combine duas partições inválidas no mesmo caso: uma falha mascara a outra.
- Cobertura = exercitados / identificados; não marque o complemento.

Essenciais para a prova (checklist)

- ☐ Sei contar partições válidas e inválidas, inclusive faixas não contínuas e listas.
- ☐ Sei calcular o mínimo de testes de EP com um e com vários campos.
- ☐ Sei listar os valores de BVA de 2 e de 3 valores, com inteiros e decimais.
- ☐ Sei montar e reduzir uma tabela de decisão e ler V/F, traço e X.
- ☐ Sei os três critérios de cobertura de transição de estado e qual representação mostra transições inválidas.
- ☐ Sei calcular cobertura de instrução e de ramificação e o mínimo de testes para IF, IF/ELSE e switch.

- ☐ Sei por que caixa-branca não acha requisitos ausentes.
- ☐ Sei distinguir suposição de erro, exploratório e lista de verificação.
- ☐ Sei os 3 Cs, o INVEST, os formatos de critério de aceite e quando o ATDD cria os testes.

Questões de revisão do capítulo

R1.1 · Análise e Modelagem de Teste

Quando os critérios de aceite de uma história devem estar definidos?

A. Antes de ela entrar no desenvolvimento
B. Só durante o teste de regressão
C. Só quando surgir um defeito
D. Só depois da implantação em produção

R1.2 · Análise e Modelagem de Teste

Ao montar uma lista de verificação para teste, qual tipo de item deve ser evitado?

A. Itens sobre usabilidade da interface
B. Itens que podem ser checados por ferramenta
C. Itens baseados em falhas comuns do domínio
D. Itens atualizados com base na experiência

R1.3 · Análise e Modelagem de Teste

A quantidade aceita vai de 1 a 99. Qual valor representa a partição inválida acima do limite?

A. Valor 100
B. Valor 1
C. Valor 50
D. Valor 99

R1.4 · Análise e Modelagem de Teste

Um empréstimo é aprovado quando o cliente tem renda comprovada e não tem restrição no nome. Qual caso deve resultar em recusa?

A. Com renda comprovada e sem restrição no nome
B. Com renda comprovada, sem restrição e aprovado
C. Sem restrição no nome e com renda comprovada
D. Com renda comprovada e com restrição no nome

Respostas comentadas

R1.1. A) Antes de ela entrar no desenvolvimento

Por quê: Critérios claros costumam fazer parte da Definição de Preparado (DoR).

✗ B) Incorreta. Seria tarde demais.
✗ C) Incorreta. Seria tarde demais.
✗ D) Incorreta. Seria tarde demais.

> **Dica de prova:** Critérios de aceite: antes do desenvolvimento.

R1.2. B) Itens que podem ser checados por ferramenta

Por quê: Também devem ser evitados itens genéricos demais e itens que funcionam melhor como critérios de entrada ou saída.

✗ A) Incorreta. São itens úteis.
✗ C) Incorreta. São itens úteis.
✗ D) Incorreta. A atualização é recomendada.

> **Dica de prova:** Lista de verificação: sem itens automáticos ou genéricos.

R1.3. A) Valor 100

Por quê: Valores abaixo de 1 formam a outra partição inválida.

✗ B) Incorreta. É o limite inferior válido.
✗ C) Incorreta. Está na partição válida.
✗ D) Incorreta. É o limite superior válido.

> **Dica de prova:** Acima do máximo: partição inválida superior.

R1.4. D) Com renda comprovada e com restrição no nome

Por quê: Regras com E só resultam na ação quando todas as condições são verdadeiras.

✗ A) Incorreta. As duas condições são atendidas.
✗ B) Incorreta. Descreve um caso aprovado.
✗ C) Incorreta. As duas condições são atendidas.

> **Dica de prova:** Regra com E: basta uma condição falsa para recusar.

02

CAPÍTULO 2 · 5% DA PROVA

Ferramentas de Teste

É o menor capítulo, 5% da prova, cerca de 2 questões, mas são pontos quase garantidos. Preciso associar cada categoria de ferramenta ao que ela faz e separar benefícios de riscos da automação. O padrão é 'qual categoria apoia tal coisa' ou 'qual destes é um risco/benefício'. Onde mais erro: gerenciamento x DevOps (rastreabilidade x pipeline) e marcar um benefício quando pedem risco, porque as alternativas misturam as duas listas.

Neste capítulo

12 tópicos-chave · 10 pegadinhas · 5 macetes · 11 gatilhos · 4 questões comentadas

Tópicos-chave

Ferramentas de gerenciamento

Apoiam o controle de requisitos, a rastreabilidade, os testes, o planejamento, o controle do processo e os relatórios de progresso.

Ferramentas de teste estático

Apoiam o processo de revisão e a análise estática. Não substituem o julgamento humano nas revisões; organizam e automatizam parte do trabalho.

Ferramentas de modelagem e implementação

Ajudam a gerar casos de teste, dados de teste e procedimentos, inclusive gerando testes a partir de modelos.

Ferramentas de execução e cobertura

Executam testes automatizados e medem a cobertura alcançada.

Ferramentas de teste não funcional

Apoiam testes como desempenho e carga, gerando carga. Algumas atividades não funcionais são inviáveis sem ferramenta, como simular milhares de usuários.

Ferramentas de DevOps

Apoiam o pipeline de entrega, a construção automatizada, a integração contínua, o fluxo de trabalho e a automação da entrega.

Ferramentas de colaboração

Facilitam a comunicação da equipe de teste.

Ferramentas de escalabilidade

Usam máquinas virtuais e contêineres para criar ambientes de teste padronizados, reduzindo problemas de configuração e acelerando o teste.

Benefícios da automação

Menos trabalho manual repetitivo, execução mais rápida e repetível, maior consistência, menos erros humanos na execução, medidas e relatórios mais objetivos, feedback mais rápido e acesso mais fácil a informações sobre testes e qualidade.

Riscos da automação

Expectativas irreais sobre a ferramenta, subestimar tempo e esforço (inclusive de manutenção), dependência excessiva da ferramenta, dependência do fornecedor, suporte insuficiente, ferramenta abandonada e incompatibilidade com a plataforma.

Dependência excessiva da ferramenta

Confiar demais na ferramenta deixa pouco espaço para o pensamento crítico; é risco, não benefício.

Automação e regressão

Testes de regressão são bons candidatos à automação porque rodam muitas vezes e mudam pouco; com integração contínua, dão feedback rápido a cada mudança. Testes de Q1 e Q4 costumam ser automatizados.

Macetes para memorizar

Categorias pelo verbo

Organiza = gerenciamento; Lê sem rodar = estático; Cria testware = modelagem; Roda e mede = execução e cobertura; Estressa = não funcional; Entrega = DevOps; Conversa = colaboração; Replica ambiente = escalabilidade.

Riscos da automação

Três E e um F: Expectativa irreal, Esforço subestimado, Excesso de dependência, Fornecedor.

Benefícios da automação

Mais rápido, mais repetível, mais consistente, mais objetivo, menos braçal.

Gerenciamento x DevOps

Gerenciamento RASTREIA; DevOps ENTREGA.

Escalabilidade

VM e contêiner = Escala o ambiente.

Tabela de memorização: não confunda

Quando é...	...e quando é
Ferramentas de gerenciamento Controle de requisitos, rastreabilidade e relatórios de progresso.	**Ferramentas de DevOps** Pipeline de entrega, construção automatizada e integração contínua.
Ferramentas de execução e cobertura Rodar testes automatizados e medir cobertura.	**Ferramentas de teste estático** Apoiar revisões e análise estática, sem executar.
Ferramentas de modelagem Gerar casos, dados e procedimentos de teste.	**Ferramentas de execução** Rodar os testes automatizados.
Ferramentas de teste não funcional Gerar carga para testes de desempenho.	**Ferramentas de escalabilidade** Criar ambientes padronizados com máquinas virtuais e contêineres.
Ferramentas de colaboração Comunicação da equipe.	**Ferramentas de gerenciamento** Planejamento, rastreabilidade e controle do processo.
Benefício da automação Menos trabalho repetitivo, consistência, medidas objetivas, feedback rápido.	**Risco da automação** Expectativas irreais, esforço subestimado, dependência da ferramenta ou do fornecedor.

Números que caem

Item	Valor
Peso do capítulo na prova	**5% (cerca de 2 questões)**

Categorias de ferramentas	**8: gerenciamento, teste estático, modelagem e implementação, execução e cobertura, não funcional, DevOps, colaboração, escalabilidade**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Marcar ferramentas de gerenciamento para pipeline e integração contínua.
Como não cair: Pipeline, build automatizado e integração contínua são ferramentas de DevOps.

⚠ **Alerta de prova:** Marcar ferramentas de DevOps para rastreabilidade e relatórios de progresso.
Como não cair: Requisitos, rastreabilidade e relatórios ficam com as ferramentas de gerenciamento.

⚠ **Alerta de prova:** Escolher teste estático ou gerenciamento para gerar carga.
Como não cair: Gerar carga é de ferramentas de teste não funcional.

⚠ **Alerta de prova:** Escolher ferramentas de modelagem para máquinas virtuais e contêineres.
Como não cair: Ambientes padronizados com VMs e contêineres são ferramentas de escalabilidade.

⚠ **Alerta de prova:** Escolher ferramentas de execução para facilitar a comunicação da equipe.
Como não cair: Comunicação é ferramenta de colaboração; execução roda os testes.

⚠ **Alerta de prova:** Escolher ferramentas de gerenciamento para gerar casos e dados de teste.
Como não cair: Gerar testware é de ferramentas de modelagem e implementação.

⚠ **Alerta de prova:** Escolher ferramentas de teste estático para medir cobertura.
Como não cair: Cobertura se mede executando: ferramentas de execução e cobertura.

⚠ **Alerta de prova:** Marcar 'dependência de um fornecedor' ou 'subestimar o esforço de manutenção' como benefício.
Como não cair: Ambos são riscos da automação.

⚠ **Alerta de prova:** Marcar 'execução mais rápida', 'mais consistência' ou 'medidas mais objetivas' como risco.
Como não cair: São benefícios; o risco típico é expectativa irreal ou dependência excessiva.

⚠ **Alerta de prova:** Achar que ferramentas de revisão substituem o julgamento humano.
Como não cair: Elas organizam e automatizam parte do trabalho; o julgamento continua humano.

Como cai na prova

📝 **Como cai:** 'Qual categoria de ferramenta apoia/faz X?'
Como resolver: Destaque o verbo ou objeto do enunciado (rastreabilidade, revisão, gerar casos, medir cobertura, carga, pipeline, comunicação, contêineres) e mapeie na categoria; elimine as que fazem outra coisa.

📝 **Como cai:** 'Qual destes é um risco da automação?' com três benefícios e um risco
Como resolver: Procure a alternativa negativa: expectativa irreal, esforço subestimado, dependência da ferramenta ou do fornecedor. As positivas são benefícios.

📝 **Como cai:** 'Qual destes é um benefício?' com três riscos e um benefício
Como resolver: Procure a alternativa positiva: menos trabalho repetitivo, consistência, medidas objetivas, feedback rápido, acesso a relatórios.

📝 **Como cai:** Pergunta sobre o efeito de depender demais de ferramentas
Como resolver: A resposta fala de pouco espaço para pensamento crítico; facilidade de relatórios e execução rápida são benefícios.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Controle de requisitos, rastreabilidade, relatórios de progresso	**Ferramentas de gerenciamento**
Processo de revisão e análise estática	**Ferramentas de teste estático**
Gerar casos, dados e procedimentos; testes a partir de modelos	**Ferramentas de modelagem e implementação**
Executar testes automatizados e medir cobertura	**Ferramentas de execução e cobertura**
Gerar carga, simular milhares de usuários	**Ferramentas de teste não funcional**
Pipeline de entrega, build automatizado, integração contínua	**Ferramentas de DevOps**
Comunicação da equipe de teste	**Ferramentas de colaboração**
Máquinas virtuais, contêineres, ambientes padronizados	**Ferramentas de escalabilidade**
Pouco espaço para o pensamento crítico	**Risco: dependência excessiva da ferramenta**
Expectativas irreais sobre a ferramenta	**Risco da automação**
Menos trabalho manual repetitivo, relatórios e medidas objetivas	**Benefício da automação**

🎯 **Se você lembrar de só uma coisa deste capítulo...**

- Gerenciamento = requisitos, rastreabilidade e relatórios; DevOps = pipeline e integração contínua.
- VMs e contêineres = escalabilidade; gerar carga = teste não funcional.
- Riscos: expectativa irreal, esforço subestimado, dependência da ferramenta e do fornecedor.

Essenciais para a prova (checklist)

- ☐ Sei associar cada uma das oito categorias de ferramentas ao que ela faz.
- ☐ Sei separar ferramentas de gerenciamento de ferramentas de DevOps.
- ☐ Sei citar benefícios da automação de teste.
- ☐ Sei citar riscos da automação de teste.
- ☐ Sei por que regressão é boa candidata à automação.
- ☐ Sei que ferramentas de revisão não substituem o julgamento humano.

Questões de revisão do capítulo

R2.1 · Ferramentas de Teste

Qual é um risco de depender demais de ferramentas de teste?

A. Pouco espaço para o pensamento crítico
B. Acesso fácil a relatórios
C. Execução mais rápida e repetível dos testes
D. Menos erros humanos na execução

R2.2 · Ferramentas de Teste

Qual categoria de ferramenta gera carga para testes de desempenho?

A. Ferramentas de teste estático
B. Ferramentas de gerenciamento
C. Ferramentas de teste não funcional
D. Ferramentas de colaboração

R2.3 · Ferramentas de Teste

Qual é um benefício da automação de teste?

A. Dependência de um fornecedor
B. Dependência excessiva da ferramenta
C. Subestimar o esforço de manutenção
D. Menos trabalho manual repetitivo

R2.4 · Ferramentas de Teste

Qual é um risco da automação de teste citado no syllabus?

A. Execução mais rápida de testes repetitivos
B. Expectativas irreais sobre a ferramenta
C. Medidas mais objetivas de cobertura
D. Maior consistência entre as execuções

Respostas comentadas

R2.1. A) Pouco espaço para o pensamento crítico

Por quê: Outros riscos: suporte insuficiente, ferramenta abandonada e incompatibilidade com a plataforma.

✗ B) Incorreta. É um benefício.
✗ C) Incorreta. É um benefício.
✗ D) Incorreta. É um benefício.

> **Dica de prova:** Risco: confiar demais na ferramenta.

R2.2. C) Ferramentas de teste não funcional

Por quê: Algumas atividades não funcionais são inviáveis sem ferramentas, como simular milhares de usuários.

✗ A) Incorreta. Apoiam revisões e análise estática.
✗ B) Incorreta. Apoiam planejamento e rastreabilidade.
✗ D) Incorreta. Apoiam a comunicação.

> **Dica de prova:** Gerar carga: ferramentas de teste não funcional.

R2.3. D) Menos trabalho manual repetitivo

Por quê: Outros benefícios: consistência, medidas objetivas, feedback mais rápido e acesso a relatórios.

✗ A) Incorreta. É um risco.
✗ B) Incorreta. É um risco.
✗ C) Incorreta. É um risco.

> **Dica de prova:** Benefício da automação: menos trabalho repetitivo.

R2.4. B) Expectativas irreais sobre a ferramenta

Por quê: Outros riscos: subestimar tempo e esforço, dependência excessiva da ferramenta e dependência do fornecedor.

✗ A) Incorreta. É um benefício.
✗ C) Incorreta. É um benefício.
✗ D) Incorreta. É um benefício.

> **Dica de prova:** Risco da automação: expectativas irreais.

03

CAPÍTULO 3 · 20% DA PROVA

Fundamentos de Teste

Aqui eu preciso dominar o vocabulário base do ISTQB: objetivos do teste, teste x depuração, erro/defeito/falha, os sete princípios, as atividades do processo de teste, testware, papéis e independência. O capítulo vale 20% da prova, cerca de 8 questões, quase sempre uma definição aplicada a um mini-cenário. O enunciado descreve uma situação e pede o princípio, a atividade ou o termo certo, e as alternativas erradas são os vizinhos do mesmo assunto. Onde eu mais erro: confundir os princípios entre si (desgaste x agrupamento x falácia), trocar análise por modelagem e chamar de falha o que é defeito. Também aparecem itens 'Escolha DUAS', em que preciso marcar exatamente as duas certas.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Objetivos do teste

Avaliar produtos de trabalho (requisitos, histórias, código), reduzir o risco de qualidade inadequada, provocar falhas e encontrar defeitos, verificar requisitos, validar o sistema e gerar confiança. O teste também fornece informação para decisões e ajuda a cumprir exigências contratuais e legais. Provar que o software não tem defeitos e corrigir defeitos NÃO são objetivos do teste.

Depuração

Atividade que localiza, analisa e remove a causa de uma falha; é distinta do teste. O teste dinâmico mostra a falha; a depuração corrige o defeito; o teste de confirmação verifica o ajuste e, depois, pode ser preciso regressão. No teste estático, a depuração se resume a remover o defeito encontrado, pois não há falha a reproduzir.

Erro, defeito e falha

Erro (engano) é a ação humana, como interpretar mal um requisito. Defeito é a imperfeição no produto de trabalho (código, requisitos, documentos). Falha é o comportamento errado observado quando o código com defeito é executado. Um defeito pode nunca causar falha, e falhas também podem vir de condições do ambiente.

Falso positivo e falso negativo

Falso positivo: o teste aponta falha, mas o problema estava nos dados, no ambiente ou no próprio testware, não no sistema. Falso negativo: o teste passa, mas o defeito existe. Defeito latente é outra coisa: um defeito que ainda não se manifestou.

Causa raiz

É o motivo fundamental que levou ao defeito, como a má interpretação de um requisito. Analisá-la serve para evitar defeitos parecidos no futuro, nunca para culpar alguém. Fatores que levam a erros: pressão de prazo, pouca experiência, complexidade, falhas de comunicação, tecnologia nova e cansaço.

Garantia da qualidade (QA) x controle da qualidade (QC)

QA é orientada ao processo e preventiva: melhorar o processo, por exemplo após auditorias. QC é orientado ao produto e corretivo; o teste é uma forma de QC. Revisar um documento para achar defeitos é teste estático, logo controle, e não QA. Resultados de teste alimentam a QA com informação sobre a eficácia dos processos.

Os sete princípios do teste

O teste mostra a presença de defeitos (não a ausência); testes exaustivos são impossíveis; testes antecipados economizam; os defeitos se agrupam; os testes se desgastam (paradoxo do pesticida); o teste depende do contexto; falácia da ausência de defeitos. A prova descreve um cenário e pede qual deles está ilustrado.

Processo de teste e contexto

Atividades: planejamento, monitoramento e controle, análise, modelagem, implementação, execução e conclusão. Elas podem ser iterativas e ocorrer em paralelo; não seguem ordem rígida. O processo é adaptado ao contexto: riscos do produto e do negócio, restrições de prazo e orçamento, partes interessadas, fatores técnicos, ciclo de vida, organização e cultura.

Planejamento, monitoramento e controle

No planejamento se definem os objetivos de teste e a abordagem para alcançá-los; ele é revisado conforme o monitoramento e o controle indicam. Monitoramento compara continuamente o progresso real com o planejado. Controle usa essas informações para tomar ações corretivas.

Análise de teste

Responde 'o que testar': examina a base de teste, identifica características testáveis e define e prioriza condições de teste. Na análise também aparecem defeitos na própria base de teste, como ambiguidades e contradições nos requisitos. Pode começar assim que a base de teste estiver disponível.

Modelagem de teste

Responde 'como testar': transforma condições de teste em casos de teste. Produz casos de teste, requisitos de dados, itens de cobertura e a definição do ambiente de teste necessário.

Implementação de teste

Responde 'tudo está pronto para executar?'. Criam-se os procedimentos de teste, organizados em suítes, e preparam-se os dados e o ambiente de teste.

Execução e conclusão

Na execução, comparam-se os resultados obtidos com os esperados, registram-se anomalias e analisam-se suas causas prováveis; dela saem os relatórios de defeito. A conclusão ocorre em marcos do projeto (fim de iteração ou release): arquiva o testware, registra lições aprendidas e gera o relatório de conclusão.

Base de teste, objeto de teste e condição de teste

Base de teste é o conjunto de documentos (requisitos, histórias de usuário) de onde se derivam os testes. Objeto de teste é o item que está sendo testado. Condição de teste é um aspecto testável do componente ou sistema e serve de base para os casos de teste e para medir cobertura.

Caso de teste e itens de cobertura

Caso de teste contém pré-condições, entradas e resultados esperados, podendo ter pós-condições e referência às condições de teste. Itens de cobertura são os elementos usados para medir a cobertura; cada técnica define os seus, como partições, valores limite, regras, transições e ramos.

Testware e rastreabilidade

Testware são os produtos de trabalho do teste: planos, condições, casos, procedimentos, dados, scripts, relatórios de defeito e de progresso; deve ficar sob gerenciamento de configuração. O código de produção é objeto de teste, não testware. A rastreabilidade entre base de teste e casos permite avaliar cobertura e impacto de mudanças e gerar relatórios compreensíveis.

Papel de gerenciamento de teste x papel de teste

Gerenciamento de teste: planejar, monitorar, controlar e coordenar a conclusão. Papel de teste: aspectos técnicos, ou seja, análise, modelagem, implementação e execução (preparar dados, projetar casos, executar e registrar falhas). Em equipes ágeis, parte das tarefas de gestão fica com a própria equipe; os papéis podem ser exercidos por pessoas diferentes ou pela mesma pessoa.

Independência, equipe inteira e habilidades

Independência cresce do próprio autor para colegas da mesma equipe, testadores da mesma empresa e testadores de fora da organização (a maior); o ideal é combinar níveis. Benefícios: reconhecer tipos diferentes de falhas e questionar premissas. Desvantagens: isolamento, desenvolvedores perderem a responsabilidade pela qualidade e testadores vistos como gargalo. Na abordagem de equipe inteira, a qualidade é de toda a equipe, e o testador complementa o desenvolvedor por buscar o que pode dar errado.

Macetes para memorizar

💡 Erro, defeito, falha

Pessoa ERRA, produto carrega o DEFEITO, execução mostra a FALHA (E → D → F, na ordem em que acontecem).

💡 Atividades técnicas

Análise = O QUÊ; Modelagem = COMO; Implementação = ESTÁ PRONTO?; Execução = RODA E COMPARA.

💡 Depois da falha

Falha → Depura → Confirma → Regride.

💡 Verificação x validação

Verificação: construímos CERTO? Validação: construímos a coisa CERTA?

💡 QA x QC

QA = Processo e Prevenção (dois P); QC/teste = Produto e Correção.

💡 Sete princípios

Palavra-chave do cenário: rigor diferente = contexto; combinações demais = exaustivo; meses sem achar nada = desgaste; poucos módulos = agrupamento; achado na revisão do requisito = antecipação; usuário rejeitou = falácia; 'está livre de defeitos' = presença.

💡 Escada de independência

Autor (zero) → colega da equipe → mesma empresa → fora da organização (máxima).

Tabela de memorização: não confunda

Quando é...	...e quando é

Falha Algo errado é observado na execução: o cliente viu o desconto errado na tela.	**Defeito** A imperfeição está no produto de trabalho: a regra de cálculo incorreta no código ou o requisito ambíguo.
Análise de teste Decidir O QUE testar: identificar condições de teste a partir da base de teste.	**Modelagem de teste** Decidir COMO testar: transformar condições em casos de teste e requisitos de dados.
Verificação Construímos certo? Comparar com a especificação escrita, o documento de projeto ou o padrão de código.	**Validação** Construímos a coisa certa? Checar com usuários se o sistema resolve o problema deles.
Garantia da qualidade (QA) Foco no processo, preventivo: melhorar o processo após auditorias.	**Teste (controle da qualidade)** Foco no produto, corretivo: executar casos, registrar falhas, revisar documentos para achar defeitos.
Os testes se desgastam Os mesmos testes repetidos há meses deixam de achar defeitos novos; é preciso atualizar testes e dados.	**Os defeitos se agrupam** A maior parte dos defeitos está concentrada em poucos módulos; isso direciona o esforço por risco.
O teste mostra a presença de defeitos Alguém conclui que o sistema está livre de defeitos porque os testes não acharam falhas.	**Falácia da ausência de defeitos** O sistema passou em tudo, sem defeitos abertos, mas os usuários o rejeitaram por não atender às necessidades.
Falso positivo O teste acusa falha, mas a causa estava nos dados de teste, no ambiente ou no testware.	**Falso negativo** O teste passa, mas o defeito real existe e não foi detectado.

Números que caem

Item	Valor
Peso do capítulo na prova	**20% (cerca de 8 questões)**
Princípios de teste	**7**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Marcar que 'o erro é o resultado observado na execução'.
Como não cair: O que se observa na execução é a falha. Erro é a ação humana que deu origem ao defeito.

⚠ **Alerta de prova:** Aceitar que toda falha vem de um defeito no código.
Como não cair: Falhas também podem vir de condições do ambiente, e nem todo resultado inesperado é falha (falso positivo).

⚠ **Alerta de prova:** Achar que defeitos existem só no código-fonte.
Como não cair: Defeitos também existem em requisitos e documentos; por isso o teste estático acha defeitos sem executar nada.

⚠ **Alerta de prova:** Escolher 'provar que o software não tem defeitos' ou 'corrigir os defeitos' como objetivo do teste.
Como não cair: O teste mostra a presença de defeitos, nunca a ausência; corrigir é depuração, normalmente feita pelo desenvolvedor.

⚠ **Alerta de prova:** Colocar o teste de confirmação antes da depuração.
Como não cair: A ordem é falha, depuração, confirmação e, se preciso, regressão. Depuração não acontece antes de qualquer teste.

⚠ **Alerta de prova:** Responder 'falácia da ausência de defeitos' quando o gerente diz que o sistema está livre de defeitos após rodadas sem falhas.
Como não cair: Esse caso é 'o teste mostra a presença de defeitos'. A falácia exige o sistema rejeitado pelos usuários por não atender às necessidades.

⚠ **Alerta de prova:** Classificar 'revisar um documento para achar defeitos' como garantia da qualidade.
Como não cair: Revisão é teste estático, uma forma de controle da qualidade. QA é melhorar o processo.

⚠ **Alerta de prova:** Dizer que a análise de teste produz os casos de teste finais ou que deve esperar o código existir.
Como não cair: Casos de teste saem da modelagem. A análise começa assim que a base de teste está disponível e também revela defeitos nela.

⚠ **Alerta de prova:** Apontar plano de teste, casos de alto nível ou condições priorizadas como produto da execução.
Como não cair: Execução gera registros e relatórios de defeito. Plano vem do planejamento, casos da modelagem, condições da análise.

⚠ **Alerta de prova:** Marcar 'mais defeitos encontrados por visões diferentes' como desvantagem da independência.
Como não cair: Isso é benefício. Desvantagens: isolamento dos testadores, desenvolvedores menos responsáveis pela qualidade, testadores vistos como gargalo, menos comunicação.

⚠ **Alerta de prova:** Escolher colegas da mesma equipe ou da mesma empresa como maior independência.
Como não cair: A maior independência é de testadores de fora da organização; o próprio autor não tem independência.

⚠ **Alerta de prova:** Afirmar que as atividades de teste seguem ordem rígida ou dispensam planejamento no ágil.
Como não cair: As atividades podem ser iterativas e paralelas, adaptadas ao contexto; o planejamento continua existindo.

⚠ **Alerta de prova:** Confundir falso positivo com defeito latente ou falso negativo.
Como não cair: Defeito latente ainda não se manifestou; falso negativo é teste que passa com defeito real. Falso positivo é falha apontada sem defeito no sistema.

⚠ **Alerta de prova:** Atribuir ao papel de teste o planejamento, o monitoramento ou a coordenação da conclusão.
Como não cair: Essas são do papel de gerenciamento de teste; o papel de teste cuida de modelar, preparar dados e executar.

Como cai na prova

📝 **Como cai:** Cenário curto seguido de 'Qual princípio de teste isso ilustra?'
Como resolver: Ache a palavra-chave do cenário (rigor, combinações, repetição, concentração, revisão cedo, usuário rejeitou, 'livre de defeitos') e mapeie no princípio; descarte os princípios que tratam de outro fenômeno.

📝 **Como cai:** História de erro humano até o problema em produção, perguntando o que é o item observado
Como resolver: Localize no tempo: ação da pessoa = erro; regra errada no código = defeito; resultado errado visto na execução = falha; motivo fundamental = causa raiz.

📝 **Como cai:** 'Em qual atividade do processo de teste...' com um verbo no enunciado
Como resolver: Use o verbo: identificar condições = análise; criar casos = modelagem; montar suítes, dados e ambiente = implementação; comparar esperado x obtido = execução; real x planejado = monitoramento; arquivar = conclusão.

📝 **Como cai:** 'Escolha DUAS' sobre erros/defeitos, objetivos ou teste x depuração
Como resolver: Marque exatamente duas. Elimine primeiro as absolutas ('toda falha', 'só no código', 'provar que não tem defeitos', 'são a mesma atividade').

📝 **Como cai:** Qual atividade é do papel de teste e qual é do gerenciamento
Como resolver: Planejar, monitorar, controlar, coordenar conclusão = gerenciamento. Preparar dados, projetar, executar, registrar falhas = papel de teste.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Dispositivo médico testado com muito mais rigor que um site de notícias	**O teste depende do contexto**
Mesmos testes de regressão há meses, sem achar defeitos novos	**Os testes se desgastam (paradoxo do pesticida)**
Maioria dos defeitos concentrada em poucos módulos	**Os defeitos se agrupam**
Milhares de combinações; a equipe usa técnicas e priorização	**Testes exaustivos são impossíveis**
Passou em todos os testes, mas os usuários rejeitaram	**Falácia da ausência de defeitos**
Defeito em requisito achado em revisão antes de qualquer código	**Testes antecipados economizam**
Problema estava nos dados de teste, não no sistema	**Falso positivo**
Identificar condições de teste a partir da base de teste	**Análise de teste**
Procedimentos organizados em suítes, dados e ambiente preparados	**Implementação de teste**
Comparar o progresso real com o planejado	**Monitoramento de teste**
Arquivar testware e registrar lições aprendidas	**Conclusão de teste**
Checar com usuários se o sistema resolve o problema deles	**Validação**
Relatar defeitos de forma construtiva / entender usuários de negócio	**Comunicação clara e boa escuta / conhecimento do domínio**
Contrato exige evidências de teste para o aceite	**O teste ajuda a cumprir exigências contratuais e legais**

🎯 **Se você lembrar de só uma coisa deste capítulo...**

- Teste mostra presença de defeitos, nunca prova ausência; corrigir é depuração, não teste.
- Defeito pode nunca virar falha, e falha pode vir do ambiente.
- Análise = o que testar; modelagem = como testar; implementação = deixar pronto para executar.
- QA é processo e preventiva; teste é produto e corretivo.
- Maior independência: testadores de fora da organização; desvantagem típica: isolamento.

Essenciais para a prova (checklist)

- ☐ Sei distinguir erro, defeito, falha e causa raiz em um cenário narrado.
- ☐ Sei reconhecer cada um dos sete princípios pela palavra-chave do cenário.
- ☐ Sei a sequência falha → depuração → confirmação → regressão.
- ☐ Sei separar verificação de validação e QA de controle da qualidade.
- ☐ Sei associar cada atividade do processo de teste aos seus produtos de trabalho.
- ☐ Sei diferenciar base de teste, objeto de teste, condição de teste e item de cobertura.
- ☐ Sei explicar falso positivo e falso negativo.
- ☐ Sei listar benefícios e desvantagens da independência de teste.
- ☐ Sei quais tarefas são do papel de teste e quais são do gerenciamento de teste.

Questões de revisão do capítulo

R3.1 · Fundamentos de Teste

Um desenvolvedor interpretou mal um requisito e escreveu uma regra de cálculo incorreta. Em produção, o sistema mostrou um desconto errado a um cliente. O desconto errado observado é:

A. Um erro (engano)
B. Um defeito
C. Uma causa raiz
D. Uma falha

R3.2 · Fundamentos de Teste

Os mesmos testes de regressão são executados há meses e deixaram de encontrar defeitos novos, embora o sistema continue mudando. Qual princípio de teste isso ilustra?

A. Os defeitos se agrupam
B. Testes antecipados economizam
C. O teste depende do contexto
D. Os testes se desgastam

R3.3 · Fundamentos de Teste

Quais elementos um caso de teste normalmente contém?

A. Entradas e resultados esperados
B. A lista de defeitos já corrigidos
C. O código-fonte da funcionalidade
D. Só o nome da funcionalidade a testar

R3.4 · Fundamentos de Teste

Qual atividade é de validação, e não de verificação?

A. Comparar o resultado com a especificação escrita
B. Checar com usuários se o sistema resolve o problema deles
C. Checar se a tela tem os campos do documento de projeto
D. Conferir se o código segue o padrão da empresa

Respostas comentadas

R3.1. D) Uma falha

Por quê: Um erro humano pode gerar um defeito no produto; quando o código com defeito é executado, pode ocorrer uma falha.

✗ A) Incorreta. O erro é a ação humana de interpretar mal o requisito.
✗ B) Incorreta. O defeito é a regra incorreta no código, que causou a falha.
✗ C) Incorreta. A causa raiz é o motivo fundamental, como a má interpretação.

> **Dica de prova:** Erro (pessoa) → defeito (produto) → falha (execução).

R3.2. D) Os testes se desgastam

Por quê: Também chamado de paradoxo do pesticida: é preciso atualizar os testes e os dados de teste.

✗ A) Incorreta. Trata da concentração de defeitos em poucos módulos.
✗ B) Incorreta. Trata do momento em que o teste começa.
✗ C) Incorreta. Trata da adaptação do teste ao contexto.

> **Dica de prova:** Testes repetidos param de achar defeitos: os testes se desgastam.

R3.3. A) Entradas e resultados esperados

Por quê: Casos de teste também podem ter pós-condições e referências às condições de teste.

✗ B) Incorreta. Não faz parte do caso de teste.
✗ C) Incorreta. Não faz parte do caso de teste.
✗ D) Incorreta. Faltam entradas e resultados.

> **Dica de prova:** Caso de teste: pré-condições, entradas, esperado.

R3.4. B) Checar com usuários se o sistema resolve o problema deles

Por quê: Verificação pergunta 'construímos certo?'; validação pergunta 'construímos a coisa certa?'.

✗ A) Incorreta. É verificação.

✗ C) Incorreta. É verificação.

✗ D) Incorreta. É verificação.

> **Dica de prova:** Validação: atende o usuário real.

04

CAPÍTULO 4 · 22% DA PROVA

Gerenciamento das Atividades de Teste

Aqui eu preciso dominar planejamento (plano de teste, critérios de entrada e saída, estimativas, priorização, pirâmide e quadrantes), risco (de produto x de projeto, análise e controle), monitoramento e controle com métricas e relatórios, gerenciamento de configuração e gerenciamento de defeitos. Vale 22% da prova, cerca de 9 questões, com algumas contas de estimativa. Os enunciados trazem uma situação de gestão e pedem a técnica, o critério ou a resposta ao risco. Onde mais erro: severidade x prioridade, extrapolação x índices, DoR x DoD (a tradução confunde) e os quadrantes Q3 x Q4.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 8 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Plano de teste

Documenta os meios e o cronograma do teste, serve de comunicação e ajuda a garantir que as atividades atendam aos critérios definidos. Inclui escopo, objetivos, riscos, abordagem, premissas, restrições, partes interessadas, comunicação, orçamento e cronograma. Não contém código, senhas nem lista de defeitos corrigidos.

Abordagem de teste

Parte do plano que descreve níveis, tipos e técnicas de teste e entregáveis, podendo incluir critérios de entrada e saída, independência e métricas.

Testador no planejamento ágil

No planejamento do release, contribui com histórias e critérios de aceite testáveis. No planejamento da iteração, analisa riscos, estima o esforço de teste, identifica aspectos não funcionais, apoia a divisão das histórias e planeja o teste delas. Não codifica nem decide sozinho o escopo.

Critérios de entrada e de saída

Entrada: disponibilidade de recursos e testware e qualidade inicial do objeto de teste (ambiente pronto e verificado, base de teste aprovada, ferramentas instaladas). Saída: medidas de cobertura, defeitos não resolvidos e riscos residuais (cobertura de requisitos atingida, defeitos críticos resolvidos). No ágil, a DoR (Definition of Ready) funciona como critério de entrada e a DoD (Definition of Done) como critério de saída.

Técnicas de estimativa

Baseada em índices (razões): proporções históricas da organização, como desenvolvimento:teste de 3:2. Extrapolação: medições das primeiras iterações do próprio projeto. Wideband Delphi: especialistas estimam separadamente e repetem rodadas até o consenso; o Planning Poker é sua variação, com cartas escondidas até todos votarem. Três pontos: otimista (a), mais provável (m) e pessimista (b).

Estimativa de três pontos

E = (a + 4m + b) / 6 e desvio-padrão = (b − a) / 6. Com a = 6, m = 9, b = 18: E = 10 e desvio = 2, apresentado como 10 ± 2. Com 4, 7 e 16 dias: E = 8 dias.

Fatores que influenciam o esforço

Produto (por exemplo, complexidade), processo, pessoas, resultados de teste e restrições. Nome do projeto ou cor da interface não influenciam.

Priorização de casos de teste

Baseada em risco (ordem do resultado da análise de risco), em cobertura (simples: maior cobertura total; adicional: o que soma mais cobertura nova) e em requisitos (prioridades definidas pelas partes interessadas). Dependências vencem a prioridade: um teste de baixa prioridade que é pré-condição roda antes.

Pirâmide de teste

Mostra mais testes na base e menos no topo: na base, testes de componente isolados, rápidos e numerosos; no topo, testes de ponta a ponta, poucos e mais lentos. Representa proporção e granularidade, não ordem de execução, e apoia automação e estimativa.

Quadrantes de teste

Q1: voltado à tecnologia e apoia a equipe (componente e integração de componentes, automatizados, na integração contínua). Q2: negócio e apoio à equipe (funcionais). Q3: negócio e crítica ao produto (exploratório, usabilidade). Q4: tecnologia e crítica ao produto (não funcionais como desempenho e segurança, exceto usabilidade). Ajudam a garantir que todos os tipos e níveis de teste sejam considerados.

Nível de risco

Determinado pela probabilidade e pelo impacto, em avaliação quantitativa ou qualitativa (matriz de risco). Alta complexidade técnica aumenta a probabilidade; equipe experiente e documentação clara tendem a reduzi-la. O impacto depende de perdas financeiras, reputação e segurança.

Risco de projeto x risco de produto

Risco de projeto afeta prazos, custos, pessoas e recursos, como atraso na entrega do ambiente de teste. Risco de produto afeta as características de qualidade, como cálculo de juros incorreto, falha de segurança no login ou tempo de resposta lento, e pode trazer insatisfação, perda de receita, danos a terceiros e custos de manutenção.

Análise de risco de produto

Identificação por brainstorming, workshops, entrevistas, diagramas de causa e efeito e listas de verificação. O resultado orienta escopo, técnicas, profundidade e prioridade do teste, além de níveis, tipos e estimativa de esforço; alimenta o planejamento, não o substitui.

Controle de risco

Inclui mitigação e monitoramento. Respostas: mitigação por teste (mais rigor e experiência, maior independência, revisões, análise estática, mais níveis de teste), aceitação (nenhuma ação), transferência (para terceiros) e plano de contingência (ações preparadas para se o risco ocorrer). O que sobra ao final é o risco residual, que ajuda a decidir a liberação.

Monitoramento, controle e métricas

Monitoramento coleta dados para avaliar o progresso; controle age sobre eles (repriorizar diante de um risco, reavaliar critérios de entrada e saída, ajustar cronograma). Métricas: progresso (casos executados e aprovados x planejados), qualidade do produto (tempo médio entre falhas, disponibilidade), defeitos (densidade por módulo, encontrados e corrigidos, severidade, percentual de detecção), risco, cobertura e custo.

Relatórios e comunicação

Relatório de progresso é periódico durante o teste (andamento, desvios, impedimentos). Relatório de conclusão sai ao fim de iteração, release ou projeto: resumo, desvios, qualidade, lições, critérios de saída, riscos residuais e métricas. O conteúdo se adapta ao público; a comunicação pode ser verbal, por painéis, canais eletrônicos ou documentos, e o gráfico de burndown mostra o trabalho restante na iteração.

Gerenciamento de configuração

Garante que itens de teste e testware estejam identificados, com versões controladas, e dá rastreabilidade entre itens, testware e resultados. Registrar a versão exata dos itens usados em um ciclo cria uma linha de base (baseline).

Relatório de defeito

Objetivo: permitir resolver o problema, acompanhar a qualidade e dar ideias de melhoria de processo. Inclui identificador, título, contexto (versão do software e ambiente), passos, resultados esperado e obtido, severidade, prioridade e status. Severidade é o grau de impacto; prioridade é a urgência de correção. Quando o comportamento está conforme a especificação, o desfecho comum é o relatório ser rejeitado (não é defeito).

Macetes para memorizar

💡 Três pontos

Média: a + 4m + b sobre 6. Desvio: b menos a sobre 6. O 'm' vale quatro.

💡 Quadrantes

Q1 e Q4 = tecnologia; Q2 e Q3 = negócio. Q1 e Q2 apoiam a equipe; Q3 e Q4 criticam o produto.

💡 Severidade x prioridade

Severidade = Sério (impacto); Prioridade = Pressa (urgência).

💡 Nível de risco

Risco = Probabilidade × Impacto.

💡 Respostas ao risco

MATC: Mitigar, Aceitar, Transferir, Contingência.

💡 DoR x DoD

Ready = pronta para ENTRAR (entrada); Done = feita para SAIR (saída).

💡 Estimativas

Índice = histórico de outros projetos; Extrapolação = medido neste projeto; Delphi = rodadas de especialistas; Três pontos = a, m, b.

💡 Risco de projeto

Projeto = prazo, custo, pessoas, ambiente; produto = qualidade do software.

Tabela de memorização: não confunda

Quando é...	...e quando é
Severidade Medir o impacto do defeito no funcionamento.	**Prioridade** Medir a urgência da correção: erro de digitação no logotipo = baixa severidade e alta prioridade.
Risco de projeto Ameaça a prazo, custo, pessoas ou ambiente: atraso na entrega do ambiente de teste.	**Risco de produto** Ameaça à qualidade do software: cálculo incorreto nas faturas, falha de segurança no login.

Critério de entrada (DoR) Para começar: ambiente pronto e verificado, base de teste aprovada, ferramentas instaladas, história preparada para a iteração.	**Critério de saída (DoD)** Para terminar: cobertura de requisitos atingida, defeitos críticos resolvidos, testes aprovados.
Extrapolação Usar medições das primeiras iterações do próprio projeto (média de esforço de três iterações).	**Estimativa baseada em índices** Usar proporções históricas de projetos anteriores da organização (desenvolvimento:teste 3:2).
Wideband Delphi Especialistas estimam separadamente e discutem em rodadas até o consenso (Planning Poker é variação).	**Estimativa de três pontos** Calcular com valores otimista, mais provável e pessimista: (a + 4m + b) / 6.
Relatório de progresso Periódico, durante o teste: andamento, desvios e impedimentos.	**Relatório de conclusão** No fim de iteração, release ou projeto: resumo, desvios, qualidade, lições e riscos residuais.
Quadrante Q3 Voltado ao negócio e crítica ao produto: exploratório e usabilidade.	**Quadrante Q4** Voltado à tecnologia e crítica ao produto: desempenho e segurança.

Números que caem

Item	Valor
Peso do capítulo na prova	**22% (cerca de 9 questões)**
Estimativa de três pontos	**E = (a + 4m + b) / 6**
Desvio-padrão na estimativa de três pontos	**(b − a) / 6**
Exemplo a = 6, m = 9, b = 18 horas	**10 ± 2 horas**
Exemplo 4, 7 e 16 dias	**8 dias**
Índice 3:2 com 600 h de desenvolvimento	**400 h de teste**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Executar primeiro o caso de alta prioridade que depende de um de baixa prioridade.
Como não cair: Dependência vence a prioridade: o de baixa roda antes, senão o outro falha sem a pré-condição.

⚠ **Alerta de prova:** Com a = 6, m = 9, b = 18, responder 10 ou 12 para o desvio-padrão.
Como não cair: 10 é a estimativa média e 12 é só b − a. Desvio = (b − a) / 6 = 2.

⚠ **Alerta de prova:** Na estimativa de três pontos, fazer a média simples dos três valores.
Como não cair: O mais provável tem peso 4: (a + 4m + b) / 6.

⚠ **Alerta de prova:** Índice 3:2 com 600 h de desenvolvimento: responder 900 h.
Como não cair: 900 inverte a proporção. Teste = 600 × 2/3 = 400 h.

⚠ **Alerta de prova:** Confundir DoR e DoD porque 'Definição de Pronto' aparece traduzindo as duas.
Como não cair: Ancore na sigla e na função: DoR (Ready) = preparada para entrar = critério de entrada; DoD (Done) = terminada = critério de saída.

⚠ **Alerta de prova:** Marcar 'defeitos críticos resolvidos' ou 'cobertura de 90% alcançada' como critério de entrada.
Como não cair: São critérios de saída. Entrada fala de recursos, testware e qualidade inicial do objeto.

⚠ **Alerta de prova:** Dizer que a pirâmide mostra a ordem de execução dos níveis ou que o topo tem os testes mais numerosos.
Como não cair: Ela mostra proporção e granularidade: muitos testes rápidos na base, poucos de ponta a ponta no topo.

⚠ **Alerta de prova:** Colocar usabilidade em Q4 junto com os não funcionais.
Como não cair: Usabilidade e exploratório ficam em Q3; Q4 tem os não funcionais técnicos, exceto usabilidade.

⚠ **Alerta de prova:** Classificar o erro de digitação no logotipo, que precisa sair antes do lançamento, como baixa severidade e baixa prioridade.
Como não cair: Impacto baixo, urgência alta: baixa severidade e alta prioridade.

⚠ **Alerta de prova:** Marcar 'corrigido e fechado' ou 'reaberto' quando o comportamento está conforme a especificação.
Como não cair: Não houve correção: o relatório é rejeitado, não é defeito (falso positivo).

⚠ **Alerta de prova:** Classificar atraso na entrega do ambiente de teste como risco de produto, ou inventar 'risco de ferramenta'.
Como não cair: Ambiente, prazo, custo e pessoas são risco de projeto. 'Risco de ferramenta' não é categoria.

⚠ **Alerta de prova:** Escolher transferência ou contingência quando a organização decide não fazer nada sobre um risco baixo.
Como não cair: Nenhuma ação é aceitação; transferir passa a terceiros; contingência prepara ações para se ocorrer.

⚠ **Alerta de prova:** Para risco de produto alto, aceitar 'reduzir cobertura', 'testar só no final' ou 'retirar do escopo'.
Como não cair: Isso aumenta o risco. Mitiga-se com mais rigor e experiência, independência, revisões, análise estática e mais níveis.

⚠ **Alerta de prova:** Chamar 'arquivar o testware' ou 'escrever o plano inicial' de ação de controle.
Como não cair: Arquivar é conclusão; plano é planejamento. Controle é repriorizar, reavaliar critérios e ajustar cronograma.

Como cai na prova

📝 **Como cai:** Valores a, m, b e pergunta de estimativa ou desvio
Como resolver: Identifique o que se pede. Estimativa: (a + 4m + b) / 6. Desvio: (b − a) / 6. Descarte a média simples, o b − a puro e o valor da estimativa no lugar do desvio.

📝 **Como cai:** Cenário de estimativa pedindo a técnica
Como resolver: Dados deste projeto = extrapolação; proporção de projetos passados = índices; especialistas em rodadas = Wideband Delphi (Planning Poker); três cenários = três pontos.

📝 **Como cai:** Situação de risco pedindo tipo ou resposta
Como resolver: Primeiro: afeta prazo/custo/pessoas (projeto) ou qualidade do software (produto)? Depois: mais teste = mitigação; nada = aceitação; passar a terceiros = transferência; plano para se ocorrer = contingência; o que sobra = residual.

📝 **Como cai:** Defeito descrito e pergunta de severidade e prioridade
Como resolver: Avalie separadamente: quanto afeta o funcionamento (severidade) e quão urgente é corrigir (prioridade). Podem ser opostos.

📝 **Como cai:** Exemplo pedindo critério de entrada ou de saída, ou métrica de certa categoria
Como resolver: Entrada = condição para começar; saída = meta atingida para terminar. Métricas: progresso (executados x planejados), qualidade (MTBF), defeitos (densidade), custo (licenças, custo total).

📝 **Como cai:** Tipo de teste e pergunta do quadrante ou da camada da pirâmide
Como resolver: Pergunte: tecnologia ou negócio? Apoia a equipe ou critica o produto? Para a pirâmide: componente na base (muitos, rápidos), ponta a ponta no topo (poucos, lentos).

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Decidir não tomar nenhuma ação sobre o risco	**Aceitação do risco**
Testar com mais profundidade as funções de maior risco	**Mitigação do risco por meio de teste**
Risco que ainda permanece ao final do teste	**Risco residual**
Média do esforço medido nas primeiras iterações	**Extrapolação**
Proporção histórica entre desenvolvimento e teste	**Estimativa baseada em índices**
Rodadas de especialistas até o consenso; cartas escondidas	**Wideband Delphi / Planning Poker**
Otimista, mais provável, pessimista	**Três pontos: (a + 4m + b) / 6**
Executar o teste que soma mais cobertura nova	**Priorização por cobertura adicional**
Ordem pela prioridade dos requisitos dada pelas partes interessadas	**Priorização baseada em requisitos**
Registrar a versão exata dos itens usados no ciclo	**Baseline (gerenciamento de configuração)**
Trabalho restante na iteração ao longo dos dias	**Gráfico de burndown**
Tempo médio entre falhas, disponibilidade	**Métrica de qualidade do produto**
Densidade de defeitos por módulo	**Métrica de defeitos**
Componente e integração de componentes / desempenho e segurança / exploratório e usabilidade	**Q1 / Q4 / Q3**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Severidade = impacto; prioridade = urgência; podem ser opostas.
- Dependência vence a prioridade na ordem de execução.
- Risco de projeto = prazo, custo, pessoas, ambiente; risco de produto = qualidade.
- Três pontos: (a + 4m + b) / 6; desvio (b − a) / 6.
- DoR = critério de entrada; DoD = critério de saída; decida pela sigla, não pela tradução.

Essenciais para a prova (checklist)

- ☐ Sei o propósito e o conteúdo do plano de teste e da abordagem de teste.
- ☐ Sei classificar exemplos como critério de entrada ou de saída, inclusive DoR e DoD.
- ☐ Sei aplicar as quatro técnicas de estimativa e calcular três pontos e desvio.
- ☐ Sei as três estratégias de priorização e o papel das dependências.
- ☐ Sei explicar a pirâmide de teste e situar tipos de teste nos quadrantes Q1 a Q4.
- ☐ Sei separar risco de projeto de risco de produto e escolher a resposta ao risco.
- ☐ Sei classificar métricas por categoria e diferenciar relatório de progresso e de conclusão.
- ☐ Sei o que é baseline e para que serve o gerenciamento de configuração.
- ☐ Sei o conteúdo de um relatório de defeito e classificar severidade e prioridade.

Questões de revisão do capítulo

R4.1 · Gerenciamento das Atividades de Teste

Depois da análise, a organização decide não tomar nenhuma ação sobre um risco de baixa prioridade. Qual resposta ao risco foi escolhida?

A. Transferência do risco
B. Plano de contingência
C. Aceitação do risco
D. Mitigação do risco

R4.2 · Gerenciamento das Atividades de Teste

Como os testadores contribuem no planejamento do release em equipes ágeis?

A. Com histórias e critérios testáveis
B. Codificando as histórias de maior valor
C. Aprovando sozinhos o escopo do release
D. Definindo o preço do produto final

R4.3 · Gerenciamento das Atividades de Teste

O que um relatório de conclusão de teste normalmente inclui?

A. Resumo, desvios, qualidade e lições
B. Contratos de fornecedores do projeto
C. Dados pessoais dos usuários de teste
D. Código-fonte das funções testadas

R4.4 · Gerenciamento das Atividades de Teste

Após três iterações, a equipe usa a média do esforço de teste medido para estimar as próximas. Qual técnica é essa?

A. Extrapolação
B. Índices históricos
C. Três pontos
D. Wideband Delphi

Respostas comentadas

R4.1. C) Aceitação do risco

Por quê: As opções de controle de risco incluem mitigação por teste, aceitação, transferência e plano de contingência.

✗ A) Incorreta. Passaria o risco para terceiros.
✗ B) Incorreta. Prepararia ações para o caso de ocorrência.
✗ D) Incorreta. Reduziria probabilidade ou impacto.

> **Dica de prova:** Nenhuma ação sobre o risco: aceitação.

R4.2. A) Com histórias e critérios testáveis

Por quê: No planejamento de iterações, testadores também detalham riscos, estimam esforço e planejam o teste das histórias.

✗ B) Incorreta. É papel do desenvolvimento.
✗ C) Incorreta. A decisão é da equipe e do negócio.
✗ D) Incorreta. Não é papel do teste.

> **Dica de prova:** Release: histórias e critérios testáveis.

R4.3. A) Resumo, desvios, qualidade e lições

Por quê: Também inclui critérios de saída, riscos residuais e métricas relevantes.

✗ B) Incorreta. Não faz parte.
✗ C) Incorreta. Não faz parte.
✗ D) Incorreta. Não faz parte.

> **Dica de prova:** Conclusão: resumo, desvios, qualidade e lições.

R4.4. A) Extrapolação

Por quê: A extrapolação é comum em projetos iterativos, com dados das primeiras iterações.

✗ B) Incorreta. Usa proporções de projetos anteriores.
✗ C) Incorreta. Usa otimista, provável e pessimista.
✗ D) Incorreta. Usa rodadas de especialistas.

> **Dica de prova:** Estimar com medições do próprio projeto: extrapolação.

05

CAPÍTULO 5 · 10% DA PROVA

Teste Estático

Aqui eu preciso dominar o que o teste estático alcança sem executar nada, o processo de revisão com suas cinco atividades, os seis papéis e os quatro tipos de revisão. Vale 10% da prova, cerca de 4 questões, e são das mais diretas: uma pergunta de 'quem faz', 'em qual atividade' ou 'que tipo de revisão é essa'. Onde eu mais erro: moderador x líder de revisão x gerente, comunicação e análise x correção e relatório, e revisão técnica x inspeção. Também preciso lembrar que tudo que exige execução (tempo de resposta, memória, carga) fica fora do estático.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 5 macetes · 13 gatilhos · 4 questões comentadas

Tópicos-chave

Teste estático

Avalia produtos de trabalho sem executá-los, por revisão manual ou análise estática com ferramenta. Encontra defeitos diretamente; o teste dinâmico provoca falhas para depois achar os defeitos. Benefício próprio: achar defeitos em itens não executáveis.

O que pode ser examinado

Quase qualquer produto de trabalho que possa ser lido e entendido: requisitos, código, planos de teste. Não é adequado o que é difícil de interpretar por pessoas ou não deve ser analisado por ferramentas, como código executável de terceiros protegido por lei.

Defeitos que o estático acha melhor

Requisitos ambíguos ou incoerentes, desvios de padrão, especificações incorretas de interface, falhas de segurança no código, variável declarada e nunca usada e código inalcançável. Lentidão sob carga, vazamento de memória e falha intermitente exigem execução.

Características avaliáveis sem execução

O estático mede características que não dependem da execução, como manutenibilidade e segurança de código. Consumo de memória, uso de processador e tempo de resposta exigem teste dinâmico.

Análise estática no pipeline

Executá-la na integração contínua detecta problemas a cada mudança, como código inalcançável e violações de padrão. Não substitui os testes de componente.

Momento das revisões

Podem ocorrer em qualquer momento do ciclo, até antes de existir código. Revisões cedo evitam que defeitos se propaguem para outros artefatos; não dependem de falhas registradas.

Feedback cedo e frequente

O feedback das partes interessadas cedo e com frequência evita mal-entendidos, mantém o foco nas funções de maior valor e reduz retrabalho. Mudanças continuam possíveis e revisões e regressão continuam necessárias.

Processo de revisão

Cinco atividades: planejamento, início da revisão, revisão individual, comunicação e análise, e correção e relatório. A prova descreve o que acontece e pede a atividade.

Planejamento e início da revisão

No planejamento definem-se escopo, objetivos, critérios, esforço e prazo. No início da revisão os documentos a revisar são distribuídos, os objetivos são explicados e garante-se que todos estejam preparados.

Revisão individual

Cada revisor examina o produto de trabalho sozinho e registra anomalias, recomendações e perguntas. É aqui que as anomalias são encontradas.

Comunicação e análise

As anomalias são discutidas e, para cada uma, decide-se status, responsável e ações. As anomalias avaliadas podem virar relatórios de defeito.

Correção e relatório

São criados os relatórios de defeito, as ações acordadas são realizadas e as correções acompanhadas. É a atividade final.

Anomalia

Na revisão, os problemas são registrados primeiro como anomalias, a confirmar depois, e não como defeitos já confirmados nem como falhas, pois não houve execução.

Papéis na revisão

Gerente decide o que revisar e fornece recursos (pessoal, tempo). Líder de revisão tem a responsabilidade geral, como decidir quem participa e onde. Moderador (facilitador) conduz a reunião com ambiente seguro. Relator (redator) registra anomalias e decisões. Autor cria e corrige. Revisor examina o produto.

Tipos de revisão

Revisão informal, acompanhamento (walkthrough), revisão técnica e inspeção. A escolha depende das necessidades e do nível de risco, e o mesmo artefato pode passar por mais de um tipo.

Revisão informal e acompanhamento

Revisão informal não tem processo nem papéis definidos nem registro formal; é comum em equipes ágeis e busca achar anomalias. Acompanhamento é conduzido pelo próprio autor e pode servir para avaliar qualidade, educar, gerar ideias e alcançar consenso.

Revisão técnica e inspeção

Revisão técnica: especialistas técnicos, liderados por um moderador, buscam consenso e decisões sobre um problema técnico, avaliam qualidade e alternativas. Inspeção é a mais formal: moderador treinado, papéis definidos, métricas coletadas e objetivo de achar o máximo de anomalias.

Fatores de sucesso da revisão

Objetivos e critérios claros, tipo de revisão adequado, revisar partes pequenas, tempo para preparação, treinamento, apoio da gestão e cultura de aprendizado. Avaliar o desempenho do autor e revisar documentos muito grandes de uma vez prejudicam.

Macetes para memorizar

💡 Atividades da revisão

Planeja → Inicia → Revisa sozinho → Comunica e analisa → Corrige e relata (P-I-R-C-C).

💡 Papéis

Gerente PAGA (recursos), Líder MANDA (responsabilidade geral), Moderador CONDUZ, Relator ANOTA, Autor CORRIGE, Revisor PROCURA.

Quem conduz cada tipo

Acompanhamento = Autor (A com A); Técnica = Técnicos; Inspeção = moderador treinado com métricas; Informal = ninguém, sem processo.

Estático x dinâmico

Estático ACHA o defeito; dinâmico PROVOCA a falha.

Fora do estático

Se precisa rodar (tempo, memória, CPU, carga), não é estático.

Tabela de memorização: não confunda

Quando é...	...e quando é
Moderador Conduzir a reunião de forma eficaz, com ambiente seguro para todos opinarem.	**Líder de revisão** Responsabilidade geral pela revisão: decidir quem participa e onde acontece.
Gerente Decidir o que será revisado e fornecer recursos como pessoal e tempo.	**Líder de revisão** Responder pela revisão como um todo e organizar participantes e local.
Relator Registrar anomalias e decisões tomadas na reunião.	**Moderador** Conduzir a reunião.
Comunicação e análise Discutir anomalias e decidir status, responsável e ações.	**Correção e relatório** Criar relatórios de defeito, realizar as ações acordadas e acompanhar correções.
Inspeção Mais formal, moderador treinado, papéis definidos, métricas, máximo de anomalias.	**Revisão técnica** Especialistas técnicos buscando consenso e decisão sobre um problema técnico, com menos formalidade.
Acompanhamento (walkthrough) Conduzido pelo autor, para educar participantes e chegar a consenso.	**Revisão informal** Colega dá uma lida rápida e manda comentários, sem processo nem registro formal.
Teste estático Achar defeitos diretamente em artefatos não executados: requisito ambíguo, variável não usada.	**Teste dinâmico** Provocar e observar falhas na execução: tempo de resposta sob carga, vazamento de memória.

Números que caem

Item	Valor
Peso do capítulo na prova	**10% (cerca de 4 questões)**

Atividades do processo de revisão	**5**
Tipos de revisão	**4: informal, acompanhamento, técnica, inspeção**
Papéis na revisão	**6: gerente, autor, moderador, relator, revisor, líder de revisão**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Registrar os problemas da revisão como 'defeitos já confirmados'.
Como não cair: Primeiro são anomalias; só depois da comunicação e análise podem virar relatórios de defeito.

⚠ **Alerta de prova:** Registrar problemas da revisão como 'falhas de execução'.
Como não cair: Revisão não executa nada; falha é coisa de teste dinâmico.

⚠ **Alerta de prova:** Achar que o estático avalia tempo de resposta, consumo de memória ou uso de processador.
Como não cair: Tudo isso exige execução. Sem execução: manutenibilidade e segurança de código.

⚠ **Alerta de prova:** Dizer que análise estática no CI substitui os testes de componente.
Como não cair: Ela dá feedback a cada mudança, mas os testes continuam necessários.

⚠ **Alerta de prova:** Aceitar que revisões só ocorrem após o teste de sistema, em produção ou quando há falhas.
Como não cair: Revisões ocorrem em qualquer momento, até sem código.

⚠ **Alerta de prova:** Escolher 'avaliar o desempenho do autor na reunião' como fator de sucesso.
Como não cair: Isso prejudica o clima e a qualidade. Sucesso vem de objetivos e critérios claros, partes pequenas, preparação, treinamento e apoio da gestão.

⚠ **Alerta de prova:** Revisar documentos grandes de uma vez ou dispensar treinamento.
Como não cair: Ambos reduzem a atenção e a eficácia dos revisores.

⚠ **Alerta de prova:** Escolher o tipo de revisão pelo número de linhas, data do projeto ou preferência do autor.
Como não cair: O tipo é escolhido conforme as necessidades e o nível de risco.

⚠ **Alerta de prova:** Dizer que o acompanhamento é conduzido por um moderador treinado.
Como não cair: Moderador treinado é da inspeção. O acompanhamento é conduzido pelo autor.

⚠ **Alerta de prova:** Marcar revisão técnica como o tipo mais formal, com métricas.
Como não cair: O mais formal, com métricas e máximo de anomalias, é a inspeção.

⚠ **Alerta de prova:** Colocar a busca de anomalias no planejamento ou no início da revisão.
Como não cair: Anomalias são achadas na revisão individual; o planejamento define escopo, objetivos e critérios; o início distribui os documentos a revisar.

⚠ **Alerta de prova:** Atribuir ao moderador a responsabilidade geral pela revisão.
Como não cair: Responsabilidade geral é do líder de revisão; o moderador conduz a reunião.

⚠ **Alerta de prova:** Achar que teste estático serve só para código compilado ou relatórios de defeito.
Como não cair: Requisitos, código e planos de teste podem ser revisados. O sistema em produção com usuários é teste dinâmico.

⚠ **Alerta de prova:** Concluir que feedback cedo garante que não haverá mudanças.
Como não cair: O benefício é evitar mal-entendidos e retrabalho; mudanças continuam possíveis.

Como cai na prova

📝 **Como cai:** 'Quem, na revisão, faz tal coisa?'
Como resolver: Use o verbo: conduz = moderador; responde por tudo e escolhe participantes = líder; dá recursos = gerente; registra = relator; cria e corrige = autor.

📝 **Como cai:** 'Em qual atividade do processo de revisão...'
Como resolver: Definir escopo e critérios = planejamento; distribuir material = início; achar anomalias = revisão individual; discutir e decidir = comunicação e análise; relatório de defeito e correções = correção e relatório.

📝 **Como cai:** Cenário de revisão pedindo o tipo
Como resolver: Pergunte quem conduz e quão formal é: autor = acompanhamento; especialistas decidindo = técnica; moderador com métricas = inspeção; sem processo = informal.

📝 **Como cai:** Benefício ou defeito típico do teste estático comparado ao dinâmico
Como resolver: Elimine toda alternativa que exige execução (carga, memória, tempo de resposta, falha intermitente); sobra o defeito em artefato não executado.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Conduzir a reunião com ambiente seguro para todos opinarem	**Moderador**
Decidir quem participa e onde a revisão acontece	**Líder de revisão**
Decidir o que revisar e fornecer pessoal e tempo	**Gerente**
Registrar anomalias e decisões da reunião	**Relator (redator)**
Cada revisor examina sozinho e anota anomalias, recomendações e perguntas	**Revisão individual**
Decidir status, responsável e ações de cada anomalia	**Comunicação e análise**
Criar relatórios de defeito e acompanhar correções	**Correção e relatório**
Mais formal, moderador, métricas, máximo de anomalias	**Inspeção**
Especialistas técnicos chegando a consenso sobre um problema técnico	**Revisão técnica**
Conduzida pelo próprio autor para educar e chegar a consenso	**Acompanhamento (walkthrough)**
Lida rápida e comentários, sem processo nem registro	**Revisão informal**
Variável declarada e nunca usada, código inalcançável	**Análise estática**
Requisito ambíguo ou incoerente	**Teste estático (revisão)**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Revisão registra anomalias, não defeitos confirmados nem falhas.
- Líder = responsabilidade geral; moderador = conduz; gerente = recursos.
- Inspeção é a mais formal; acompanhamento é conduzido pelo autor.
- Tudo que exige execução fica fora do teste estático.

Essenciais para a prova (checklist)

- ☐ Sei citar defeitos que o teste estático encontra melhor que o dinâmico.
- ☐ Sei quais produtos de trabalho podem e quais não podem passar por teste estático.

- ☐ Sei as cinco atividades do processo de revisão e o que acontece em cada uma.
- ☐ Sei os seis papéis da revisão e a função de cada um.
- ☐ Sei diferenciar os quatro tipos de revisão por quem conduz e pela formalidade.
- ☐ Sei os fatores de sucesso de uma revisão e o que a prejudica.
- ☐ Sei por que rodar análise estática na integração contínua.

Questões de revisão do capítulo

R5.1 · Teste Estático

Em qual atividade do processo de revisão são criados relatórios de defeito e acompanhadas as correções?

A. Revisão individual
B. Planejamento
C. Correção e relatório
D. Início da revisão

R5.2 · Teste Estático

Em qual atividade do processo de revisão as anomalias são discutidas, e se decide o status, o responsável e as ações de cada uma?

A. Planejamento
B. Comunicação e análise
C. Revisão individual
D. Início da revisão

R5.3 · Teste Estático

O que é definido no planejamento de uma revisão?

A. O relatório final da revisão
B. Escopo, objetivos e critérios
C. As anomalias encontradas
D. A correção das anomalias

R5.4 · Teste Estático

Qual benefício é próprio do teste estático, quando comparado ao teste dinâmico?

A. Observar falhas durante a execução
B. Avaliar o tempo de resposta sob carga
C. Achar defeitos em itens não executáveis
D. Medir o desempenho do sistema em uso

Respostas comentadas

R5.1. C) Correção e relatório

Por quê: As atividades são planejamento, início, revisão individual, comunicação e análise, e correção e relatório.

✗ A) Incorreta. Cada revisor examina o produto sozinho.
✗ B) Incorreta. Define escopo, objetivos e critérios.
✗ D) Incorreta. Distribui o material e explica os objetivos.

> **Dica de prova:** Relatórios e correções: correção e relatório.

R5.2. B) Comunicação e análise

Por quê: Na correção e relatório, são criados relatórios de defeitos e as ações acordadas são realizadas.

✗ A) Incorreta. Define escopo e critérios.
✗ C) Incorreta. Cada revisor encontra anomalias.
✗ D) Incorreta. Prepara os participantes.

> **Dica de prova:** Discutir e decidir anomalias: comunicação e análise.

R5.3. B) Escopo, objetivos e critérios

Por quê: O planejamento também define o esforço e o prazo da revisão.

✗ A) Incorreta. É feito ao final.
✗ C) Incorreta. São achadas na revisão individual.
✗ D) Incorreta. Ocorre depois.

> **Dica de prova:** Planejamento da revisão: escopo, objetivos, critérios.

R5.4. C) Achar defeitos em itens não executáveis

Por quê: O teste estático encontra defeitos diretamente; o dinâmico provoca falhas para depois achar os defeitos.

✗ A) Incorreta. É característica do teste dinâmico.
✗ B) Incorreta. Exige execução do sistema.
✗ D) Incorreta. Exige execução do sistema.

> **Dica de prova:** Estático: defeitos em artefatos não executados.

06

CAPÍTULO 6 · 15% DA PROVA

Teste ao Longo do Ciclo de Vida de Desenvolvimento

Aqui eu preciso saber como o ciclo de vida muda o teste, as abordagens de teste primeiro (TDD, ATDD, BDD), DevOps e shift-left, os cinco níveis de teste, os tipos de teste e o teste de manutenção. Vale 15% da prova, cerca de 6 questões. O padrão é um cenário curto pedindo o nível, a forma de aceite ou o tipo de teste. Onde eu mais erro: integração de componentes x integração de sistemas, aceite operacional x aceite do usuário, alfa x beta e confirmação x regressão. Também preciso lembrar que tipos de teste cabem em qualquer nível.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Influência do ciclo de vida

O ciclo de vida escolhido afeta escopo, momento, documentação, técnicas e automação do teste. Em modelos sequenciais o teste dinâmico tende a vir tarde; em iterativos e incrementais cada iteração tem teste estático e dinâmico, documentação mais leve e muito teste de regressão, de preferência automatizado.

Boas práticas em qualquer ciclo

Cada atividade de desenvolvimento tem uma atividade de teste correspondente; cada nível de teste tem objetivos próprios; a análise e a modelagem começam na fase correspondente; testadores participam cedo das revisões de rascunhos.

TDD

Desenvolvimento orientado a testes: o primeiro passo de cada ciclo é escrever um teste que falha; depois o código para fazê-lo passar; por fim, a refatoração. É abordagem de teste primeiro.

BDD

Desenvolvimento orientado a comportamento: os comportamentos esperados são descritos em linguagem natural no formato Dado/Quando/Então. Os cenários podem ser traduzidos em testes automatizados executáveis e vêm antes do código.

Abordagens de teste primeiro

TDD, ATDD e BDD aplicam shift-left, criando testes antes do código. Os testes criados podem ser mantidos como regressão automatizada; essas abordagens costumam aumentar a automação e não substituem revisões de requisitos.

DevOps e o teste

Benefícios: feedback rápido com testes no pipeline, menos erros em ambientes de teste e mais regressão automatizada. Riscos: esforço para manter o pipeline e a automação, necessidade de habilidades em ferramentas e de infraestrutura. Testadores, ambientes e teste manual continuam necessários.

Shift-left

Antecipar o teste no ciclo. Exemplos: revisar requisitos antes da codificação, TDD, integração contínua, análise estática antes do teste dinâmico e teste não funcional desde cedo.

Retrospectivas

Trazem melhoria contínua do processo e dos testes, melhor qualidade do testware, colaboração e vínculo entre desenvolvimento e teste. Não garantem ausência de defeitos nem substituem aceite ou planejamento.

Níveis de teste

Componente, integração de componentes, sistema, integração de sistemas e aceite. Um nível se distingue do outro por objeto de teste, objetivos, base de teste, defeitos e falhas típicos e abordagem; nunca pela ferramenta ou pelo número de testadores.

Teste de componente

Testa unidades isoladas, geralmente com stubs e drivers e estruturas de teste, no ambiente de desenvolvimento, normalmente pelos próprios desenvolvedores.

Teste de integração de componentes

Foca nas interações e interfaces entre componentes já testados isoladamente; a base típica são as especificações de interface. A integração incremental (de cima para baixo ou de baixo para cima, seguindo a arquitetura) é preferida ao big bang porque facilita isolar a causa das falhas; stubs e drivers continuam úteis.

Teste de sistema

Avalia o sistema inteiro, de ponta a ponta, incluindo características funcionais e não funcionais, em ambiente parecido com o de produção, feito pela equipe do projeto.

Teste de integração de sistemas

Testa as interfaces do sistema com outros sistemas, ambientes externos e serviços de terceiros, como um serviço externo de pagamento.

Teste de aceite e suas formas

O objetivo principal é validar que o sistema está pronto para uso; achar defeitos não é o foco. Formas: aceite do usuário (UAT, usuários finais em ambiente que simula produção), operacional (administradores: backup, restauração, recuperação de desastres, instalação, usuários, segurança operacional), contratual (critérios do contrato), regulatório (normas legais obrigatórias), alfa e beta (software de prateleira).

Teste funcional x não funcional

Funcional avalia o que o sistema deve fazer (cupom, frete, gravação no banco). Não funcional avalia quão bem ele se comporta, pelas características da ISO/IEC 25010: eficiência de desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade. O não funcional deve começar o mais cedo possível.

Caixa-preta x caixa-branca como tipos

Caixa-preta deriva testes das especificações externas; caixa-branca, da estrutura interna do sistema. Funcional, não funcional, caixa-preta e caixa-branca podem ser usados em qualquer nível de teste.

Confirmação e regressão

Confirmação reexecuta o teste que falhou para verificar a correção. Regressão verifica se a mudança não quebrou partes que já funcionavam; a análise de impacto ajuda a definir seu escopo. Regressão é boa candidata à automação porque roda muitas vezes e muda pouco.

Teste de manutenção

Disparado por mudanças no sistema em produção: modificações, atualizações, migrações (por exemplo para nova plataforma) e aposentadoria. O escopo depende do risco e do tamanho da mudança, definido com análise de impacto. Na aposentadoria podem ser necessários testes de migração e arquivamento de dados e de restauração dos dados arquivados.

Macetes para memorizar

💡 Os cinco níveis

Peça → encaixe das peças → máquina inteira → máquina conversando com outras → cliente aprova: componente, integração de componentes, sistema, integração de sistemas, aceite.

💡 Alfa x beta

Alfa = Aqui, na casa do fornecedor; Beta = Bem longe, na casa do cliente.

💡 TDD

Teste que falha → código que passa → refatora.

💡 Confirmação x regressão

Confirmação olha o conserto; regressão olha os vizinhos do conserto.

💡 Funcional x não funcional

Funcional = O QUÊ; não funcional = QUÃO BEM.

💡 Gatilhos de manutenção

MAMA: Modificação, Atualização, Migração, Aposentadoria.

💡 Formas de aceite

U-O-C-R + A-B: Usuário, Operacional, Contratual, Regulatório, Alfa, Beta.

Tabela de memorização: não confunda

Quando é...	...e quando é
Teste de confirmação Reexecutar o caso que havia falhado para verificar a correção em si.	**Teste de regressão** Verificar se as partes que já funcionavam continuam funcionando após a mudança (efeitos colaterais).
Integração de componentes Interfaces internas entre componentes já testados isoladamente; base: especificações de interface.	**Integração de sistemas** Interfaces com outros sistemas, ambientes externos e serviços de terceiros.
Teste alfa Clientes testam nas instalações da empresa desenvolvedora, antes do lançamento.	**Teste beta** Clientes em potencial testam no próprio ambiente, antes do lançamento ao mercado.
Aceite operacional Administradores e operação testam backup, restauração, recuperação de desastres, instalação e gerenciamento de usuários.	**Aceite do usuário (UAT)** Usuários finais validam se o sistema atende às suas necessidades em ambiente que simula a produção.
Aceite contratual Verificar critérios previstos em contrato.	**Aceite regulatório** Verificar conformidade com normas governamentais e leis obrigatórias.

Teste funcional O que o sistema faz: o cupom aplica o desconto, o frete está certo.	**Teste não funcional** Quão bem faz: tempo de resposta com muitos usuários, facilidade de instalação.
Teste de componente Unidade isolada, com stubs e drivers, feita pelos desenvolvedores.	**Teste de sistema** Sistema inteiro de ponta a ponta, funcional e não funcional, em ambiente parecido com produção.

Números que caem

Item	Valor
Peso do capítulo na prova	**15% (cerca de 6 questões)**
Níveis de teste	**5: componente, integração de componentes, sistema, integração de sistemas, aceite**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Chamar de teste de componente ou de integração de componentes o teste com um serviço externo de pagamento.
Como não cair: Interface com outro sistema ou serviço de terceiros é integração de sistemas.

⚠ **Alerta de prova:** Aceitar que caixa-branca ocorre só no nível de componente, não funcional só no aceite ou regressão só no sistema.
Como não cair: Tipos de teste cabem em qualquer nível; desconfie de 'só' nessas alternativas.

⚠ **Alerta de prova:** Dizer que ciclos iterativos precisam de menos regressão ou de documentação mais formal.
Como não cair: Mudanças frequentes aumentam a regressão, de preferência automatizada; a documentação costuma ser mais leve.

⚠ **Alerta de prova:** Colocar 'escrever o código de produção' ou 'refatorar' como primeiro passo do TDD.
Como não cair: O ciclo começa com um teste que falha; código depois; refatoração no fim.

⚠ **Alerta de prova:** Marcar 'feedback mais rápido' ou 'mais regressão automatizada' como risco do DevOps.
Como não cair: Esses são benefícios. O risco é o esforço de manter o pipeline e a automação, além de habilidades e infraestrutura.

⚠ **Alerta de prova:** Acreditar que DevOps acaba com o teste manual, os ambientes de teste ou o papel dos testadores.
Como não cair: Tudo isso continua necessário; DevOps acelera o feedback no pipeline.

⚠ **Alerta de prova:** Dizer que integração incremental elimina stubs ou dispensa teste de componente.
Como não cair: A vantagem é isolar falhas com facilidade; stubs, drivers e teste de componente continuam úteis. Integrar tudo de uma vez é big bang.

⚠ **Alerta de prova:** Escolher 'encontrar defeitos' como objetivo principal do aceite.
Como não cair: O aceite valida que o sistema está pronto para uso; ele pode revelar defeitos, mas esse não é o objetivo principal.

⚠ **Alerta de prova:** Na aposentadoria de um legado, escolher usabilidade, carga ou componente.
Como não cair: O foco é migração e arquivamento dos dados e a restauração dos dados arquivados.

⚠ **Alerta de prova:** Achar que o escopo do teste de manutenção depende do número de testadores ou da idade do sistema.
Como não cair: Depende do risco e do tamanho da mudança, avaliados por análise de impacto.

⚠ **Alerta de prova:** Marcar a primeira versão em desenvolvimento como gatilho de teste de manutenção.
Como não cair: Manutenção é para sistema já em produção: modificação, atualização, migração e aposentadoria.

⚠ **Alerta de prova:** Deixar o teste não funcional para depois do aceite ou para a produção.
Como não cair: Começa o mais cedo possível; muitos testes não funcionais derivam de funcionais e ocorrem em vários níveis.

⚠ **Alerta de prova:** No modelo sequencial, aceitar 'deixar todo o teste para depois da codificação' ou 'mesmos objetivos em cada nível'.
Como não cair: A análise começa na fase correspondente, cada nível tem objetivos próprios e testadores entram cedo nas revisões.

⚠ **Alerta de prova:** Confundir caixa-branca com especificações externas ou com experiência do testador.
Como não cair: Caixa-branca vem da estrutura interna; especificações externas são caixa-preta; experiência é base das técnicas baseadas em experiência.

Como cai na prova

📝 **Como cai:** Cenário descreve quem testa, onde e o quê, e pede o nível de teste
Como resolver: Identifique o objeto: unidade isolada = componente; interface interna = integração de componentes; sistema inteiro = sistema; outro sistema/terceiro = integração de sistemas; prontidão para uso = aceite.

📝 **Como cai:** Cenário pede a forma de teste de aceite
Como resolver: Veja quem testa e contra o quê: usuário e necessidades = UAT; administradores e backup = operacional; contrato = contratual; lei/norma = regulatório; local do fornecedor = alfa; ambiente do cliente = beta.

📝 **Como cai:** Após uma correção, qual teste fazer
Como resolver: Se é o mesmo caso que falhou, confirmação; se é verificar o resto do sistema, regressão; se o sistema está em produção e mudou, é teste de manutenção com análise de impacto.

📝 **Como cai:** Benefício ou risco de DevOps, shift-left ou teste primeiro
Como resolver: Benefícios falam de feedback rápido e automação; riscos falam de manter pipeline, habilidades e infraestrutura. Alternativas que 'eliminam' teste manual, testadores ou revisões estão erradas.

📝 **Como cai:** Implicação de um modelo de ciclo de vida para o teste
Como resolver: Sequencial: teste dinâmico tarde, mas análise começa na fase correspondente. Iterativo: teste em cada iteração, mais regressão automatizada, documentação mais leve.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Unidades isoladas, stubs e drivers, executado pelos desenvolvedores	**Teste de componente**
Interfaces entre componentes já testados isoladamente	**Teste de integração de componentes**
Interface com serviço externo ou sistema de terceiros	**Teste de integração de sistemas**
Sistema inteiro, de ponta a ponta	**Teste de sistema**
Backup, restauração, recuperação de desastres feitos por administradores	**Aceite operacional**
Normas governamentais obrigatórias	**Aceite regulatório**
Clientes nas instalações do fornecedor / no próprio ambiente	**Alfa / beta**
Reexecutar o caso que havia falhado após correção	**Teste de confirmação**
Partes que já funcionavam continuam funcionando	**Teste de regressão**

Avaliar quais áreas podem ser afetadas antes de mudar o sistema em produção	**Análise de impacto**
Migração para nova plataforma ou aposentadoria	**Teste de manutenção**
Tempo de resposta com muitos usuários simultâneos	**Eficiência de desempenho (não funcional)**
Dado/Quando/Então em linguagem natural	**BDD**
Revisar requisitos antes de codificar, análise estática antes do dinâmico	**Shift-left**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Tipos de teste cabem em qualquer nível; alternativa com 'só em tal nível' costuma estar errada.
- Confirmação verifica a correção; regressão procura efeitos colaterais.
- Integração de componentes = interfaces internas; integração de sistemas = outros sistemas e terceiros.
- Escopo de manutenção = risco e tamanho da mudança, via análise de impacto.
- TDD começa com um teste que falha.

Essenciais para a prova (checklist)

- ☐ Sei reconhecer os cinco níveis de teste por objeto, base e responsável.
- ☐ Sei distinguir as seis formas de teste de aceite pelo cenário.
- ☐ Sei separar teste de confirmação de teste de regressão.
- ☐ Sei classificar um exemplo como funcional ou não funcional e citar características da ISO/IEC 25010.
- ☐ Sei a ordem do ciclo de TDD e o formato do BDD.
- ☐ Sei citar benefícios e riscos do DevOps para o teste.
- ☐ Sei dar exemplos de shift-left.
- ☐ Sei os gatilhos do teste de manutenção e o que testar na aposentadoria.
- ☐ Sei como o ciclo de vida sequencial ou iterativo muda o teste.

Questões de revisão do capítulo

R6.1 · Teste ao Longo do Ciclo de Vida de Desenvolvimento

Um sistema legado será aposentado, e seus dados precisam ser guardados por anos. Que teste de manutenção pode ser necessário?

A. Teste de usabilidade da nova interface
B. Teste de carga com usuários simulados
C. Teste de componente das funções antigas
D. Teste de migração e arquivamento de dados

R6.2 · Teste ao Longo do Ciclo de Vida de Desenvolvimento

De onde os testes de caixa-branca são derivados?

A. Do cronograma do projeto
B. Das especificações externas
C. Da experiência do testador
D. Da estrutura interna do sistema

R6.3 · Teste ao Longo do Ciclo de Vida de Desenvolvimento

Medir o tempo de resposta do sistema com muitos usuários simultâneos avalia qual característica de qualidade?

A. Adequação funcional
B. Manutenibilidade
C. Eficiência de desempenho
D. Portabilidade do sistema

R6.4 · Teste ao Longo do Ciclo de Vida de Desenvolvimento

Do que depende o escopo do teste de manutenção?

A. Da cor da interface do sistema
B. Do número de testadores disponíveis
C. Da data de criação do sistema
D. Do risco e do tamanho da mudança

Respostas comentadas

R6.1. D) Teste de migração e arquivamento de dados

Por quê: Na aposentadoria, podem ser necessários testes de migração de dados e de restauração e recuperação dos dados arquivados.

✗ A) Incorreta. Não se aplica a um sistema que sai de uso.
✗ B) Incorreta. Não é o foco na aposentadoria.
✗ C) Incorreta. Não é o foco na aposentadoria.

> **Dica de prova:** Aposentadoria: testar migração e arquivamento.

R6.2. D) Da estrutura interna do sistema

Por quê: Caixa-branca e caixa-preta podem ser aplicados em qualquer nível de teste.

✗ A) Incorreta. Não é base para derivar testes.
✗ B) Incorreta. Essa é a base do teste de caixa-preta.
✗ C) Incorreta. É a base das técnicas baseadas em experiência.

> **Dica de prova:** Caixa-branca: estrutura interna.

R6.3. C) Eficiência de desempenho

Por quê: As características não funcionais da ISO/IEC 25010 incluem desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade.

✗ A) Incorreta. Trata da correção das funções.
✗ B) Incorreta. Trata da facilidade de mudança.
✗ D) Incorreta. Trata da troca de ambiente.

> **Dica de prova:** Tempo de resposta sob carga: eficiência de desempenho.

R6.4. D) Do risco e do tamanho da mudança

Por quê: A análise de impacto ajuda a definir o escopo do teste de manutenção.

✗ A) Incorreta. Não influencia.
✗ B) Incorreta. Influencia a capacidade, não o escopo necessário.
✗ C) Incorreta. Não influencia diretamente.

> **Dica de prova:** Escopo de manutenção: risco e tamanho da mudança.

DIA DA PROVA

Estratégia de prova

A prova tem 40 questões de múltipla escolha em 60 minutos (75 para quem faz em idioma não nativo), e a aprovação exige 26 acertos (65%). As questões são curtas: um mini-cenário seguido de 'qual princípio', 'qual atividade', 'qual nível', 'qual técnica' ou 'qual papel', com alternativas erradas que são conceitos vizinhos do mesmo capítulo. Análise e Modelagem de Teste (28%) e Gerenciamento das Atividades de Teste (22%) somam metade da prova e trazem contas: valores limite, partições, regras de tabela de decisão, cobertura de instrução e ramificação, estimativa de três pontos. Alguns itens pedem 'Escolha DUAS'. O vocabulário é o do syllabus CTFL v4.0, e a tradução às vezes usa nomes parecidos para coisas diferentes, como DoR e DoD.

Como responder

✓ Distribua o tempo: 60 minutos para 40 questões dá cerca de 1,5 minuto por questão; responda as de definição rápido e guarde tempo para as de cálculo.

✓ Foque o estudo pelo peso: Análise e Modelagem (cerca de 11 questões) e Gerenciamento (cerca de 9) primeiro, depois Fundamentos (8), Ciclo de Vida (6), Teste Estático (4) e Ferramentas (2).

✓ Em todo cenário, sublinhe a palavra-chave (carta de teste, stubs e drivers, backup, normas obrigatórias, nenhuma ação sobre o risco) e só então leia as alternativas.

✓ Nas contas, escreva a faixa ou a lógica numa linha: limites e vizinhos, partições, ramos ou a, m e b; confira o passo da precisão e se a partição é aberta.

✓ Desconfie de alternativas absolutas: 'provar que não há defeitos', 'só no nível de componente', 'elimina o teste manual', 'substitui as revisões' quase sempre estão erradas.

✓ Elimine alternativas de outro capítulo: cobertura de ramificação como resposta para técnica de experiência, estimativa de três pontos para identificar riscos.

✓ Em 'Escolha DUAS', marque exatamente duas e comece eliminando as falsas absolutas.

✓ Lembre da margem: com 26 de 40 você pode errar até 14; não trave numa questão difícil, marque e volte.

Erros que derrubam candidatos

✗ Marcar o percentual não coberto (20%, 25%) no lugar da cobertura.

✗ Usar o vizinho interno na BVA de 2 valores ou esquecer um vizinho na de 3 valores.

✗ Achar que 100% de cobertura de instrução garante 100% de ramificação.

✗ Trocar confirmação por regressão e integração de componentes por integração de sistemas.

✗ Confundir severidade com prioridade e risco de projeto com risco de produto.

✗ Escolher a técnica de estimativa pelo nome mais conhecido em vez do dado usado (projeto atual, histórico, especialistas, três valores).

✗ Confundir moderador, líder de revisão e gerente nos papéis da revisão.

Na semana da prova

- ☐ Refaça os cálculos de BVA, EP, tabela de decisão, cobertura de instrução e ramificação e três pontos até sair sem pensar.
- ☐ Revise os sete princípios, as atividades do processo de teste e os papéis da revisão pelas palavras-chave dos cenários.
- ☐ Faça simulados completos cronometrados em 60 minutos e analise cada erro pela alternativa que te enganou.
- ☐ Revise as tabelas de comparação: confirmação x regressão, alfa x beta, Q1 a Q4, DoR x DoD, severidade x prioridade.
- ☐ Na véspera, só revisão leve dos macetes e números; nada de conteúdo novo.

REVISÃO FINAL

Folha de cola

Leia na véspera e na manhã da prova: os gatilhos e as pegadinhas mais importantes de cada domínio.

1. Análise e Modelagem de Teste

‣ Carta de teste, sessão com tempo, debriefing → **Teste exploratório baseado em sessão**

‣ Falhas que já viu em sistemas parecidos; divisão por zero; ataques de falha → **Suposição de erro**

‣ Lista de itens atualizada com a experiência de projetos anteriores → **Teste baseado em lista de verificação**

‣ Dado / Quando / Então → **Critério de aceite orientado a cenário (e BDD)**

‣ Testes escritos a partir dos critérios antes de implementar a história → **ATDD**

⚠ 100% é a cobertura de instrução; de ramificação são 50%, pois o ramo falso não foi exercitado.

⚠ É o inverso: ramificação 100% leva a instrução 100%. O IF sem ELSE derruba a afirmação contrária.

⚠ 4 seria todas as combinações. Como são independentes, 2 testes (tudo verdadeiro, tudo falso) cobrem os ramos.

2. Ferramentas de Teste

‣ Controle de requisitos, rastreabilidade, relatórios de progresso → **Ferramentas de gerenciamento**

‣ Processo de revisão e análise estática → **Ferramentas de teste estático**

‣ Gerar casos, dados e procedimentos; testes a partir de modelos → **Ferramentas de modelagem e implementação**

‣ Executar testes automatizados e medir cobertura → **Ferramentas de execução e cobertura**

‣ Gerar carga, simular milhares de usuários → **Ferramentas de teste não funcional**

⚠ Pipeline, build automatizado e integração contínua são ferramentas de DevOps.

⚠ Requisitos, rastreabilidade e relatórios ficam com as ferramentas de gerenciamento.

⚠ Gerar carga é de ferramentas de teste não funcional.

3. Fundamentos de Teste

‣ Dispositivo médico testado com muito mais rigor que um site de notícias → **O teste depende do contexto**

‣ Mesmos testes de regressão há meses, sem achar defeitos novos → **Os testes se desgastam (paradoxo do pesticida)**

‣ Maioria dos defeitos concentrada em poucos módulos → **Os defeitos se agrupam**

‣ Milhares de combinações; a equipe usa técnicas e priorização → **Testes exaustivos são impossíveis**

‣ Passou em todos os testes, mas os usuários rejeitaram → **Falácia da ausência de defeitos**

⚠ O que se observa na execução é a falha. Erro é a ação humana que deu origem ao defeito.

⚠ Falhas também podem vir de condições do ambiente, e nem todo resultado inesperado é falha (falso positivo).

⚠ Defeitos também existem em requisitos e documentos; por isso o teste estático acha defeitos sem executar nada.

4. Gerenciamento das Atividades de Teste

‣ Decidir não tomar nenhuma ação sobre o risco → **Aceitação do risco**

‣ Testar com mais profundidade as funções de maior risco → **Mitigação do risco por meio de teste**

‣ Risco que ainda permanece ao final do teste → **Risco residual**

‣ Média do esforço medido nas primeiras iterações → **Extrapolação**

‣ Proporção histórica entre desenvolvimento e teste → **Estimativa baseada em índices**

⚠ Dependência vence a prioridade: o de baixa roda antes, senão o outro falha sem a pré-condição.

⚠ 10 é a estimativa média e 12 é só b − a. Desvio = (b − a) / 6 = 2.

⚠ O mais provável tem peso 4: (a + 4m + b) / 6.

5. Teste Estático

‣ Conduzir a reunião com ambiente seguro para todos opinarem → **Moderador**

‣ Decidir quem participa e onde a revisão acontece → **Líder de revisão**

‣ Decidir o que revisar e fornecer pessoal e tempo → **Gerente**

‣ Registrar anomalias e decisões da reunião → **Relator (redator)**

‣ Cada revisor examina sozinho e anota anomalias, recomendações e perguntas → **Revisão individual**

⚠ Primeiro são anomalias; só depois da comunicação e análise podem virar relatórios de defeito.

⚠ Revisão não executa nada; falha é coisa de teste dinâmico.

⚠ Tudo isso exige execução. Sem execução: manutenibilidade e segurança de código.

6. Teste ao Longo do Ciclo de Vida de Desenvolvimento

‣ Unidades isoladas, stubs e drivers, executado pelos desenvolvedores → **Teste de componente**

‣ Interfaces entre componentes já testados isoladamente → **Teste de integração de componentes**

‣ Interface com serviço externo ou sistema de terceiros → **Teste de integração de sistemas**

‣ Sistema inteiro, de ponta a ponta → **Teste de sistema**

‣ Backup, restauração, recuperação de desastres feitos por administradores → **Aceite operacional**

⚠ Interface com outro sistema ou serviço de terceiros é integração de sistemas.

⚠ Tipos de teste cabem em qualquer nível; desconfie de 'só' nessas alternativas.

⚠ Mudanças frequentes aumentam a regressão, de preferência automatizada; a documentação costuma ser mais leve.

GABARITO

Respostas do diagnóstico

D1. B) 80% → revise o capítulo 1

Por quê: Cobertura de instrução é o número de instruções executadas dividido pelo total de instruções executáveis.

✗ A) Incorreta. Não corresponde ao cálculo.

✗ C) Incorreta. É o percentual não executado.

✗ D) Incorreta. Duas instruções não foram executadas.

> **Dica de prova:** Cobertura de instrução: executadas / total.

D2. D) 8 regras → revise o capítulo 1

Por quê: Com n condições booleanas, a tabela completa tem 2 elevado a n regras.

✗ A) Incorreta. Seria 3 × 3.

✗ B) Incorreta. Seria uma por condição.

✗ C) Incorreta. Seria o dobro do número de condições.

> **Dica de prova:** Tabela completa: 2^n regras.

D3. B) Cobertura de 75% → revise o capítulo 1

Por quê: A cobertura de BVA é o número de valores limite exercitados dividido pelo total identificado.

✗ A) Incorreta. É a parte não coberta.

✗ C) Incorreta. Cálculo incorreto.

✗ D) Incorreta. Cálculo incorreto.

> **Dica de prova:** Cobertura BVA: exercitados / identificados.

D4. A) Ferramentas de gerenciamento → revise o capítulo 2

Por quê: Outras categorias incluem ferramentas de modelagem, execução e cobertura, não funcionais e de colaboração.

✗ B) Incorreta. Apoiam o pipeline de entrega.

✗ C) Incorreta. Geram carga e medem desempenho.

✗ D) Incorreta. Apoiam revisões e análise estática.

> **Dica de prova:** Requisitos, rastreabilidade e relatórios: ferramentas de gerenciamento.

D5. B) Ferramentas de teste estático → revise o capítulo 2

Por quê: As ferramentas não substituem o julgamento humano nas revisões; elas ajudam a organizar e automatizar parte do trabalho.

✗ A) Incorreta. Apoiam a comunicação.
✗ C) Incorreta. Apoiam o pipeline de entrega.
✗ D) Incorreta. Apoiam desempenho e carga.

> **Dica de prova:** Revisão e análise estática: ferramentas de teste estático.

D6. B) Ferramentas de DevOps → revise o capítulo 2

Por quê: Ferramentas de DevOps também acompanham fluxo de trabalho e automação da entrega.

✗ A) Incorreta. Apoiam revisões.
✗ C) Incorreta. Apoiam a comunicação.
✗ D) Incorreta. Apoiam a criação de testes.

> **Dica de prova:** Pipeline e integração contínua: ferramentas de DevOps.

D7. A) QA foca no processo; teste avalia o produto → revise o capítulo 3

Por quê: O controle da qualidade é orientado ao produto e corretivo; a garantia da qualidade é orientada ao processo e preventiva.

✗ B) Incorreta. São atividades diferentes.
✗ C) Incorreta. QA acompanha todo o processo.
✗ D) Incorreta. É o inverso.

> **Dica de prova:** QA: processo; teste (QC): produto.

D8. C) Ela também avalia a base de teste → revise o capítulo 3

Por quê: Na análise, a base de teste é examinada para identificar características testáveis e definir condições de teste; defeitos na base também aparecem aqui.

✗ A) Incorreta. A análise pode começar assim que a base de teste está disponível.
✗ B) Incorreta. Casos de teste são produzidos na modelagem.
✗ D) Incorreta. Complementa, mas não substitui, as revisões.

> **Dica de prova:** Análise de teste: o que testar e defeitos na base.

D9. C) Depuração do código → revise o capítulo 3

Por quê: Teste dinâmico mostra falhas; depuração encontra e corrige os defeitos que as causam.

✗ A) Incorreta. Verifica se a correção funcionou.
✗ B) Incorreta. Verifica efeitos colaterais de mudanças.
✗ D) Incorreta. Identifica o que testar.

> **Dica de prova:** Encontrar e corrigir a causa: depuração.

D10. B) Densidade de defeitos por módulo → revise o capítulo 4

Por quê: Outras métricas de defeitos: encontrados e corrigidos, severidade e percentual de detecção.

✗ A) Incorreta. É métrica de preparação.
✗ C) Incorreta. Não é métrica de teste.
✗ D) Incorreta. É métrica de custo.

> **Dica de prova:** Métrica de defeitos: densidade.

D11. C) 2 horas → revise o capítulo 4

Por quê: A estimativa média é (a + 4m + b) / 6 = 10 horas, e o resultado costuma ser apresentado como 10 ± 2.

✗ A) Incorreta. Não corresponde à fórmula.
✗ B) Incorreta. É a estimativa média, não o desvio.
✗ D) Incorreta. É só a diferença b − a.

> **Dica de prova:** Desvio-padrão em três pontos: (b − a) / 6.

D12. D) Arquivar o testware e as lições → revise o capítulo 4

Por quê: A conclusão ocorre em marcos do projeto, como o fim de uma iteração ou release, e gera o relatório de conclusão.

✗ A) Incorreta. Faz parte da modelagem.
✗ B) Incorreta. Faz parte da implementação.
✗ C) Incorreta. Faz parte da análise.

> **Dica de prova:** Fim do ciclo: arquivar e registrar lições.

Agora, treine no formato real

Faça as 6 provas completas do curso ISTQB CTFL: 240 questões inéditas, tempo do exame, explicação de cada alternativa e link da documentação para revisar.

Ver as 6 provas completas e o cupom do mês →

Até o dia da prova, uma revisão por dia no seu feed

Instagram @strecht.simulados
Um post de prova por dia: questão comentada, pegadinhas e o "não confunda" de cada certificação.
Seguir no Instagram →

YouTube Strecht Simulados
Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.
Inscrever-se no canal →

Material independente de preparação, sem afiliação ou endosso de ISTQB. As questões são inéditas e não reproduzem a prova real.