**📘** Resumo - Preparação para CTFL 4.0

Introdução

Informações sobre o Exame:

* 40 questões (cada uma vale 1 ponto)
* Duração de 60 minutos
* Para passar: 65% de acerto
* Níveis K - (K1 - Lembrar, K2- Entender, K3 - Aplicar)
* Gabarito não é divulgado
* Glossário de termos: <https://glossary.istqb.org/>
* ISO 29119, 25010, 20246

## **1. Fundamentos do Teste de Software**

### **🔹 O que é teste de software?**

Teste é o processo de **avaliar um sistema ou seus componentes** para verificar se ele atende aos requisitos especificados.

### **🔹 Objetivos principais:**

* Encontrar defeitos (bugs).
* Verificar se o sistema está conforme especificações.
* Garantir qualidade e confiança no software.

### **🔹 Diferença entre erro, defeito e falha:**

| **Termo** | **Definição** |
| --- | --- |
| **Erro (Error)** | Ação humana incorreta que leva a um defeito. Ex: lógica errada no código. |
| **Defeito (Defect)** | Imperfeição no software. Ex: variável não inicializada. |
| **Falha (Failure)** | Quando o software se comporta de forma inesperada. Ex: sistema trava. |

### **🔹 Sete princípios do teste:**

1. Teste mostra presença de defeitos, não sua ausência.
2. Testes exaustivos são impossíveis.
3. Teste antecipado economiza tempo e dinheiro.
4. Agrupamento de defeitos (poucos módulos concentram muitos bugs).
5. Paradoxo do pesticida (testes repetidos deixam de encontrar novos bugs).
6. Teste depende do contexto (ex: testes para sistemas bancários ≠ jogos).
7. A ausência de erros não garante que o software é útil.

## **2. Processo de Teste**

### **🔹 Ciclo de vida do teste (Test Process):**

1. **Planejamento de Teste (Test Planning)** – Definir escopo, recursos e cronograma.
2. **Monitoramento e Controle de Testes** – Acompanhar progresso e ajustar planos.
3. **Análise e Design de Testes** – Identificar o que será testado e como.
4. **Implementação e Execução** – Criar e executar os casos de teste.
5. **Avaliação de critérios de saída** – Verificar se os objetivos foram cumpridos.
6. **Encerramento dos Testes** – Finalizar atividades, documentar lições aprendidas.

## **Revisões:**

1. **Revisão Informal**: leitura sem registro formal.
2. **Revisão Walkthrough**: apresentação informal com feedback.
3. **Revisão Técnica**: equipe de especialistas analisa.
4. **Inspeção**: processo formal, documentado e com papéis definidos.

# **📈 Tipos de Teste**

* **Funcionais**: requisitos funcionais.
* **Não funcionais**: desempenho, usabilidade, segurança.
* **Re-teste**: executar novamente após correção.
* **Teste de Regressão**: verifica se não quebrou o que já funcionava.

## **3. Teste Estático**

### **🔹 O que é?**

São testes realizados **sem executar o código**.

### **🔹 Tipos:**

* **Revisões** – Análise de documentos (ex: especificações, código, plano de teste).
* **Análise Estática** – Feita por ferramentas que detectam erros automaticamente.

## **4. Técnicas de Teste (Dinâmico)**

### **🔹 Técnicas baseadas em:**

#### **➤ Caixa Preta (sem olhar o código)**

* **Particionamento de equivalência** – Dividir entradas em grupos válidos/ inválidos.
* **Análise do valor limite** – Testar limites (ex: idade mínima permitida = 18).
* **Tabela de decisão** – Regras de negócio com múltiplas condições.
* **Transição de estados** – Testar mudanças de estado (ex: login/logout).

#### **➤ Caixa Branca (olhando o código)**

* Cobertura de instruções e decisões.
* Exemplo: verificar se todos os caminhos "if/else" são executados.

#### **➤ Baseada em experiência**

* **Teste exploratório** – O testador decide o que testar com base na experiência.
* **Teste baseado em erros anteriores**.

## **5. Gerenciamento de Teste**

### **🔹 Tarefas do Gerente de Testes:**

* Planejar, monitorar e controlar.
* Gerenciar riscos e comunicação.

### **🔹 Riscos:**

* **Produto**: bugs graves, segurança, desempenho.
* **Projeto**: prazos, orçamento, ferramentas ineficazes.

### **🔹 Métricas:**

* Casos de teste executados, defeitos encontrados, cobertura de requisitos.

## **6. Ferramentas de Teste**

### **🔹 Tipos de ferramentas:**

| **Tipo** | **Exemplo** |
| --- | --- |
| **Gerenciamento de Teste** | TestRail, Zephyr |
| **Automação de Testes** | Selenium, JUnit |
| **Teste de Performance** | JMeter |
| **Análise Estática** | SonarQube |
| **Gerenciamento de Defeitos** | Jira, Bugzilla |

## **7. Teste em Ciclos de Vida de Software**

### **🔹 Modelos de Desenvolvimento:**

| **Modelo** | **Característica** |
| --- | --- |
| **Cascata** | Etapas sequenciais (requisitos → desenvolvimento → testes). |
| **V-Model** | Validação e verificação lado a lado. |
| **Ágil** | Iterativo e incremental. Ex: Scrum. |

### **🔹 Níveis de teste:**

1. **Teste de Unidade** – feito por desenvolvedores (testa funções/classes).
2. **Teste de Integração** – valida interação entre componentes.
3. **Teste de Sistema** – testa o sistema completo.
4. **Teste de Aceitação** – feito pelo cliente (ex: UAT).

## **8. Papel do Testador em Equipes Ágeis**

### **🔹 Participação ativa:**

* Colaborar em histórias de usuário.
* Automatizar testes (CI/CD).
* Participar de reuniões diárias (Scrum).

### **🔹 Técnicas úteis:**

* **BDD (Behavior-Driven Development)** – Ex: Given/When/Then.
* **TDD (Test-Driven Development)** – Escreve testes antes do código.

## **🔑 Principais Siglas e Termos**

| **Sigla** | **Significado** | **Explicação** |
| --- | --- | --- |
| **CTFL** | Certified Tester Foundation Level | Certificação nível básico. |
| **ISTQB** | International Software Testing Qualifications Board | Entidade certificadora. |
| **TDD** | Test-Driven Development | Teste antes da codificação. |
| **BDD** | Behavior-Driven Development | Foco no comportamento do sistema. |
| **UAT** | User Acceptance Testing | Teste de aceitação pelo usuário. |
| **CI/CD** | Continuous Integration / Continuous Deployment | Entregas automatizadas e frequentes. |

| **ATDD** | Acceptance Test Driven Development | Testes de aceitação antes do código |
| --- | --- | --- |