CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

OLÁ!

Eu sou o Leonardo Carvalho

QA Engineer | Trabalho com QA desde 2008

Pós-graduado em Eng. de Software | CTFL | CTFL-AT | CTAL-TM

youtube.com/@leointech

@leogdcarvalho

linkedin.com/in/leogcarvalho

ÍNDICE

SOBRE O EXAME

40 QUESTÕES / 1 PONTO POR QUESTÃO

60 MINUTOS

MODALIDADE ONLINE E PRESENICIAL

65% DE ACERTO PARA PASSAR (26 QUESTÕES)

NÍVEIS K (K1: lembrar • K2: entender • K3: aplicar)

O GABARITO NÃO É DIVULGADO

VISÃO GERAL

DISTRIBUIÇÃO DE TEMPO / QUESTÕES

Ficha do Exame

Pré-requisitos: nenhum
Idioma: Língua Portuguesa (Brasil)
Número de questões: 40
Tipo de questões: múltipla escolha
Tempo de Exame: 60 minutos (75 min. para estrangeiros)
Pontuação: 40 pontos (1 ponto por questão)
Aprovação: mínimo de 65% de acertos ou 26 pontos
Distribuição das questões e pontuações:

Capítulos	1º	2º	3º	4º	5º	6º
Questões	8	6	4	11	9	2
Pontuação	8	6	4	11	9	2

MUITA COISA VOCÊ VAI TER QUE DECORAR

O CAPÍTULO 4 É O MAIS DIFÍCIL

O SYLLABUS É CANSATIVO E ÀS VEZES DIFÍCIL DE ENTENDER

(Mas sem ler ele inteiro, vai ser difícil você passar)

APESAR DE MUITO TEÓRICO, TUDO QUE VOCÊ VAI ESTUDAR É POSSÍVEL DE APLICAR!

MAS AGORA, FOQUE EM ESTUDAR PARA PASSAR NA PROVA.

MATERIAL OFICIAL: SYLLABUS + GLOSSÁRIO + SIMULADOS

MATERIAL COMPLEMENTAR: ESSE CURSO / AUDIO SYLLABUS / QUIZZES / SIMULADOS / ETC

VISÃO GERAL

SELOS USADOS NO MATERIAL PARA TE AJUDAR

INDICA O NÍVEL K DAQUELE CONTEÚDO

NÍVEIS K (K1: lembrar • K2: entender • K3: aplicar)

INDICA QUE VOCÊ PRECISA DECORAR O QUE ESTÁ NAQUELE SLIDE

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

CAPÍTULO 1

Fundamentos de Teste

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

1.1 O QUE É TESTE?

FL-1.1.1 (K1) Identificar os objetivos típicos de teste
FL-1.1.2 (K2) Diferenciar o teste da depuração

1.1 O QUE É TESTE?

Softwares que não funcionam

Todos já tivemos uma experiência com software que não funcionou como esperado.

Consequências

- Dinheiro
- Tempo
- Reputação
- Ferimentos / Morte

TESTE DE SOFTWARE

avalia a qualidade *do software e ajuda a* ***reduzir o risco de falha*** *do software em operação.*

1.1 O QUE É TESTE?

NÃO É

- Apenas executar o software e verificar resultados
- Apenas verificar requisitos (se o sistema atende aos requisitos especificados)

É

- Planejamento
- Análise
- Modelagem
- Implementação
- Execução
- Relatórios (progresso e resultados)
- Validar se atende às necessidades dos usuários

TESTE DE SOFTWARE

*deve estar **alinhado com o ciclo de vida de desenvolvimento** de software.*

1.1 O QUE É TESTE?

MAIS CARACTERÍSTICAS

- Atividades para descobrir defeitos
- Atividades para avaliar a qualidade dos artefatos de software
- O teste não só verifica (se atende aos requisitos), mas também valida (se atende as necessidades.
- Apesar de existirem muitas ferramentas que auxiliam o teste, o teste é em grande parte uma atividade intelectual (exige habilidades analíticas, pensamento crítico e sistêmico)

1.1.1 OBJETIVOS DO TESTE

ENCONTRAR DEFEITOS

EVITAR DEFEITOS

CRIAR CONFIANÇA SOBRE O NÍVEL DE QUALIDADE

FORNECER INFORMAÇÕES PARA TOMADA DE DECISÃO

OUTROS

- Avaliar produtos de trabalho
- Garantir a cobertura necessária do objeto de teste
- Reduzir nível de risco
- Verificar conformidade com requisitos

DECORAR!

K1

OBJETIVOS DO TESTE

*podem **variar** de acordo com o*

contexto, nível de teste e modelo de ciclo de vida

1.1.2 TESTE E DEPURAÇÃO

TESTE

Mostra falhas causadas por defeitos (bugs) no software.

Confirma se um defeito foi corrigido (Teste de confirmação)

Regressão para verificar se tudo continua funcionando

DEPURAÇÃO (DEBUG)

Reproduzir a falha, Diagnosticar (encontrar a causa), e Corrigir.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

1.2 POR QUE OS TESTES SÃO NECESSÁRIOS?

FL-1.2.1 (K2) Exemplificar por que os testes são necessários
FL-1.2.2 (K1) Relembrar a relação entre testes e garantia de qualidade
FL-1.2.3 (K2) Distinguir entre causa raiz, erro, defeito e falha

1.2 POR QUE O TESTE É NECESSÁRIO?

Testes ajudam a **atingir objetivos acordados dentro do escopo, tempo, qualidade e restrições orçamentárias**.

Todos podem testar de acordo com suas habilidades para colaborar com o sucesso do projeto.

1.2.1 CONTRIBUIÇÕES DO TESTE PARA O SUCESSO

- Testes contribuem para sistemas de maior qualidade pois detectam defeitos e esses podem ser removidos
- O teste avalia a qualidade de um sistema em diversos momentos do SDLC, ajudando em decisões
- Os testes também podem ser necessários para atender a requisitos contratuais ou legais, ou normas regulatórias.
- Testadores precisam entender as necessidades dos usuários e considerar isso dentro do ciclo de vida de desenvolvimento.

1.2.2 GARANTIA DA QUALIDADE E TESTE

- Teste e QA não são a mesma coisa. O teste é uma forma de controle de qualidade (QC).
- QC é uma abordagem corretiva e orientada para o produto que se concentra nas atividades que apoiam a obtenção de níveis adequados de qualidade.

QA (Quality Assurance)

- QA é uma abordagem preventiva e orientada para o processo, se concentra na implementação e aprimoramento dos processos (se um bom processo for seguido corretamente, ele gerará um bom produto).

- É responsabilidade de todos em um projeto

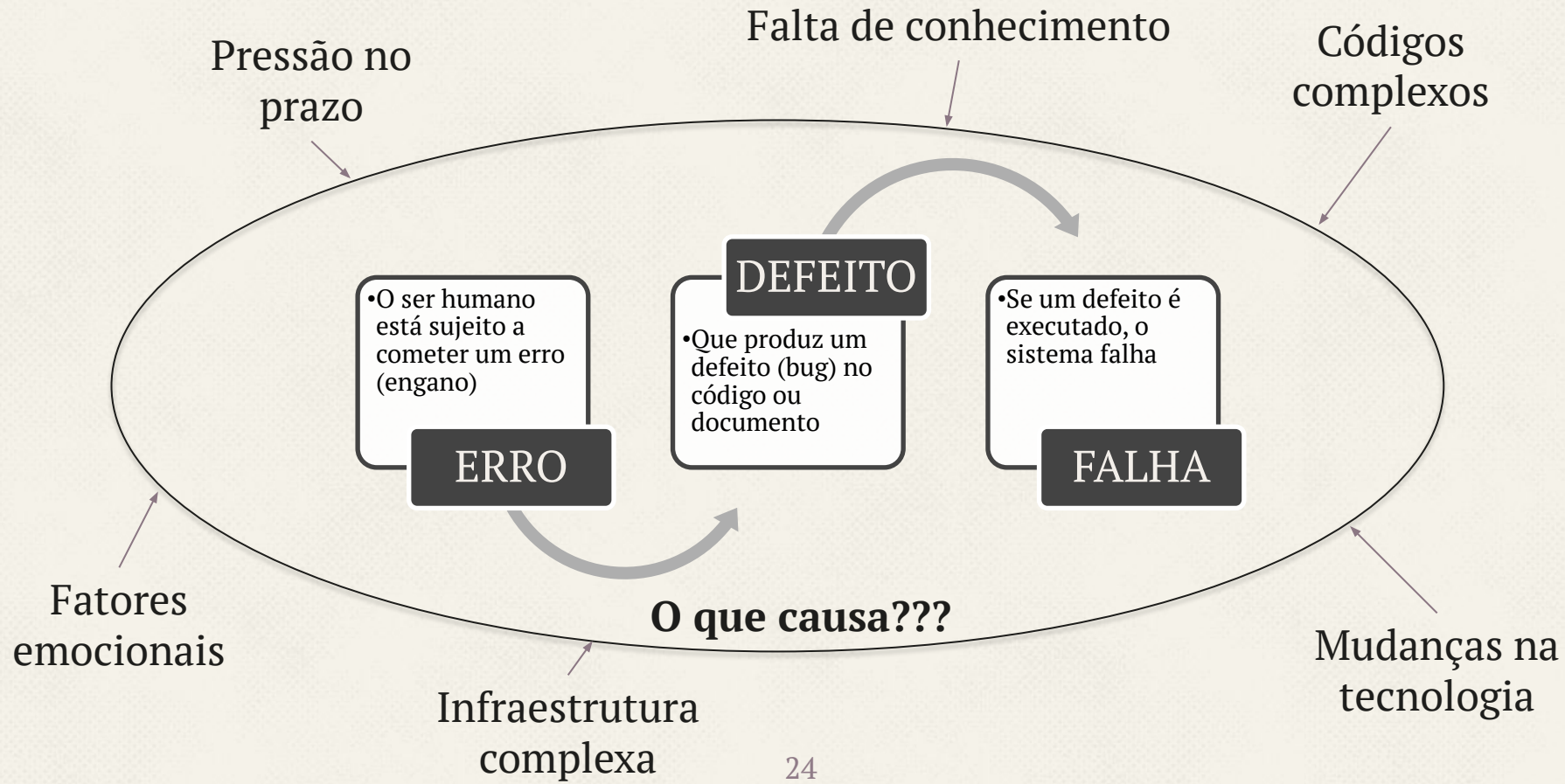
1.2.2 GARANTIA DA QUALIDADE E TESTE

- Os **resultados dos testes são usados por QA e QC**.
- No QC, eles são usados para **corrigir defeitos**
- No QA eles fornecem **feedback sobre a performance dos processos** de desenvolvimento e teste.

1.2.3 ERROS, DEFEITOS E FALHAS E CAUSAS-RAIZ

DECORAR!

1.2.3 ERROS, DEFEITOS E FALHAS E CAUSAS-RAIZ



1.2.3 ERROS, DEFEITOS E FALHAS E CAUSAS-RAIZ

- Alguns defeitos **sempre resultarão em falha** se forem executados, enquanto outros só resultarão em falhas em **circunstâncias específicas**, e alguns podem **nunca** resultar em falha.
- As falhas também podem ser causadas por condições **ambientais** (Clima, campo eletromagnético, etc)

CAUSAS-RAIZ

- **Motivo fundamental** para a ocorrência de um problema
- São identificadas por meio da **análise de causa-raiz**, realizada quando ocorre uma falha ou quando um defeito é identificado
- Analisando e remediando a causa-raiz, é provável que outras falhas ou **defeitos semelhantes sejam evitados futuramente**

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

1.3 OS 7 PRINCÍPIOS DE TESTE

FL-1.3.1 (K2) Explicar os sete princípios de teste

1.3 OS SETE PRINCÍPIOS DO TESTE

Princípio 1 – O teste mostra a presença, não a ausência de defeitos

- O teste pode demonstrar a presença de defeitos, mas não pode provar que eles não existem.

Princípio 2 – Testes exaustivos são impossíveis

- Testar tudo não é viável.

Princípio 3 – Testes antecipados economizam tempo e dinheiro

- A atividade de teste deve começar o mais breve possível no ciclo de desenvolvimento.

DECORAR!

Princípio 4 – Defeitos se agrupam

- Um número pequeno de funções contém a maioria dos defeitos

Princípio 5 – Os testes se degradam

- Pode ocorrer de um mesmo conjunto de testes que são repetidos várias vezes não encontrarem novos defeitos após um determinado momento. Os casos de testes necessitam ser frequentemente revisados e atualizados.

Princípio 6 – Os testes dependem do contexto

- Testes são realizados de forma diferente conforme o contexto.

Princípio 7 – Falácia da ausência de defeitos

- Encontrar e consertar defeitos não ajuda se o sistema construído não atende às expectativas dos usuários.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

1.4 ATIVIDADES DE TESTE, TESTWARE E PAPÉIS NO TESTE

FL-1.4.1 (K2) Resumir as diferentes atividades e tarefas de teste
FL-1.4.2 (K2) Explicar o impacto do contexto no processo de teste
FL-1.4.3 (K2) Diferenciar testware que dá suporte às atividades de teste
FL-1.4.4 (K2) Explicar o valor de manter a rastreabilidade
FL-1.4.5 (K2) Comparar os diferentes papéis no teste

1.4 PROCESSO DE TESTE

Não existe um processo universal de teste

Conjunto de atividades para maior probabilidade de atingir os objetivos

Processo de teste pode ser adaptado a uma determinada situação com base em vários fatores

O Processo ideal depende de muitos fatores.

PROCESSO DE TESTE

depende do Contexto!

1.4.1 ATIVIDADES E TAREFAS DE TESTE

Um processo de teste possui as seguintes principais atividades

Planejamento do Teste

Monitoramento e Controle do Teste

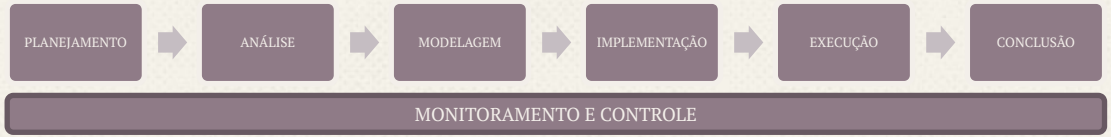
Análise do Teste

Modelagem do Teste

Implementação do Teste

Execução do Teste

Conclusão do Teste



DECORAR!

***Essas atividades podem ocorrer em paralelo**

PLANEJAMENTO DO TESTE

Envolve as atividades que **definem os objetivos e a abordagem do teste** para atender aos objetivos do teste dentro das restrições impostas pelo contexto.

Os planos de teste podem ser revisitados com base no feedback das atividades de monitoramento e controle.

MONITORAMENTO E CONTROLE

Monitoramento: **Verificação** contínua de todas as atividades de teste e **comparação** do progresso real com o plano de teste.

Controle: **Tomada de ações** necessárias para atender aos objetivos do teste.

O progresso é comunicado aos Stakeholders por meio de relatórios

***Mais detalhes no capítulo 5.3**

ANÁLISE DO TESTE

Determina “O que” testar

Analisa a **base de teste apropriada ao nível de teste** (especificações, modelagens, código, etc), para identificar r**ecursos testáveis** e definir e priorizar **condições de teste e riscos.**

A base de teste e os objetos de teste **também são avaliados para identificar defeitos** que possam conter e para avaliar sua testabilidade. Geralmente é apoiada pelo uso de técnicas de teste (cap 4).

MODELAGEM DO TESTE

Determina “Como” testar

- **Transforma condições de teste em casos de teste** e outros materiais (ex. cartas de teste).
- **Identificação de itens de cobertura**, que servem como guia para especificar as entradas do caso de teste.
- **Técnicas de teste (cap 4) podem ser usadas** para apoiar a modelagem.
- Também inclui definição dos requisitos de **dados de teste**, projeto do **ambiente de teste** e a identificação de qualquer outra infraestrutura e ferramenta necessária.

IMPLEMENTAÇÃO DO TESTE

- **Criação ou a aquisição do material de teste** necessário para a execução (ex. dados de teste).
- Casos de teste podem ser **organizados em procedimentos de teste** (passos) e geralmente são reunidos em conjuntos de testes.
- São **criados scripts de teste manuais e automatizados**.
- Procedimentos de teste são **priorizados e organizados em um cronograma** de execução de teste.
- O **ambiente de teste é criado** e verificado quanto à configuração correta.

EXECUÇÃO DO TESTE

- **Executar/rodar os testes** de acordo com o cronograma de execução.
- A execução do teste pode ser **manual ou automatizada**.
- Pode assumir várias formas, inclusive **testes contínuos** ou sessões de **testes em pares**.
- **Resultados atuais** dos testes são **comparados** com os **resultados esperados**.
- Resultados do teste são **registrados**.
- As anomalias (bugs) são analisadas para identificar suas **causas** prováveis. Essa análise permite **relatar** as anomalias com base nas falhas observadas.

CONCLUSÃO DO TESTE

- **Ocorrem nos marcos** do projeto (Ex: lançamento, fim da iteração, conclusão do nível de teste).
- Qualquer **material de teste que possa ser útil** no futuro é identificado e arquivado ou entregue às equipes apropriadas.
- O **ambiente de teste** é encerrado em um estado acordado.
- As atividades de teste são analisadas para **identificar as lições aprendidas** e as melhorias futuras.
- Um r**elatório de conclusão do teste** é criado e comunicado aos stakeholders.

1.4.2 PROCESSO DE TESTE NO CONTEXTO

As atividades de teste são **parte integrante dos processos de desenvolvimento**.

Os testes são **financiados pelos stakeholders** e seu objetivo final é ajudar a atender às **necessidades de negócio deles**.

1.4.2 PROCESSO DE TESTE NO CONTEXTO

Fatores do contexto que influenciam o teste:

- **Stakeholders** (necessidades, expectativas, requisitos, disposição para cooperar etc.).
- **Membros da equipe** (habilidades, conhecimento, nível de experiência, disponibilidade, necessidades de treinamento etc.).
- **Domínio do negócio** (criticidade do objeto de teste, riscos identificados, necessidades do mercado, normas legais específicas etc.).
- **Fatores técnicos** (tipo de software, arquitetura do produto, tecnologia usada etc.).
- **Restrições do projeto** (escopo, tempo, orçamento, recursos etc.).
- **Fatores organizacionais** (estrutura organizacional, políticas existentes, práticas utilizadas etc.).
- **Ciclo de vida do desenvolvimento de software** (práticas de engenharia, métodos de desenvolvimento etc.).
- **Ferramentas** (disponibilidade, usabilidade, conformidade etc.)

1.4.2 PROCESSO DE TESTE NO CONTEXTO

Esses fatores impactarão:

- Estratégia de teste
- Técnicas de teste usadas
- Grau de automação de teste
- Nível de cobertura necessária
- Nível de detalhe da documentação de teste
- Relatórios
- etc.

1.4.3 TESTWARE

Testware são produtos de trabalho de saída de cada atividade de teste.

Cada organização produz, molda, nomeia, organiza e gerencia seus produtos de trabalho de uma forma diferente.

A gestão de configuração (ref 5.4) garante a consistência e a integridade dos produtos de trabalho.

1.4.3 TESTWARE

Planejamento do Teste

- Planos de Teste
  - Cronograma de testes
  - Registro de riscos
  - Critérios de entrada e saída

Monitoramento e Controle

- Relatórios de progresso de testes
- Documentação de diretrizes de controle
- Informações sobre Riscos

1.4.3 TESTWARE

Análise

- Condições de Teste definidas e priorizadas
- Relatórios de Defeitos não corrigidos diretamente

Modelagem

- Casos de Teste
- Carta de Teste
- Dados de Teste, Cobertura, Ambientes

1.4.3 TESTWARE

Implementação

- Procedimentos de Teste (Passo a passo / Detalhamento)
- Scripts de Teste automatizado
- Conjuntos de Teste / Dados de Teste
- Cronograma de Execução
- Elementos do Ambiente de Teste (drivers, stubs, virtualização)

Execução

- Registros de Teste
- Relatórios de Defeitos

Conclusão

- Relatórios de Teste
- Itens de ação / lições aprendidas documentadas para melhoria dos próximos projetos

1.4.4 RASTREABILIDADE ENTRE A BASE DE TESTE E O TESTWARE

Para implementar o monitoramento e o controle eficazes dos testes, é importante estabelecer e manter a r**astreabilidade em todo o processo de teste**.

BASE DE TESTE ⟷ TESTWARE ⟷ RESULTADOS DOS TESTES ⟷ DEFEITOS

1.4.4 RASTREABILIDADE ENTRE A BASE DE TESTE E O TESTWARE

A rastreabilidade precisa dá suporte à avaliação da cobertura, que é um indicador muito importante.

- A **rastreabilidade dos casos de teste aos requisitos** pode verificar se os requisitos são cobertos pelos casos de teste.
- A **rastreabilidade dos resultados dos testes aos riscos** pode ser usada para avaliar o nível de risco residual em um objeto de teste.

1.4.4 RASTREABILIDADE ENTRE A BASE DE TESTE E O TESTWARE

- Uma boa rastreabilidade permite determinar o **impacto das mudanças**, facilita as auditorias de teste e ajuda a atender os critérios de governança de TI.

- A boa rastreabilidade também **torna o progresso do teste e os relatórios de conclusão mais compreensíveis**.

- A rastreabilidade **fornece informações para avaliar a qualidade do produto**, a capacidade do processo e o progresso do projeto em relação aos objetivos de negócio.

1.4.5 PAPÉIS NO TESTE

Gerenciamento de Teste

- Responsabilidade geral pelo processo de teste, pela equipe de teste e pela liderança das atividades de teste
- Atividades de planejamento, monitoramento e controle, e conclusão de testes
- Papel de gerenciamento de testes varia de acordo com o contexto.
- No desenvolvimento ágil, algumas das tarefas de gerenciamento de testes podem ser realizadas pela equipe ágil, e as tarefas que abrangem várias equipes ou toda a organização podem ser executadas por gerentes de teste fora da equipe de desenvolvimento.

Testador

- Responsabilidade geral pelo aspecto de engenharia (técnico) do teste
- **Atividades de análise, modelagem, implementação e execução de teste.**
- **Executar os testes**, avaliar os resultados e documentar os desvios dos resultados esperados.
- **Automatizar** os testes conforme necessário
- **Avaliar as características não funcionais**, como eficiência de desempenho, confiabilidade, usabilidade, segurança, compatibilidade e portabilidade.
- Dependendo dos riscos relacionados ao produto e ao projeto, e o modelo de ciclo de vida selecionado, **pessoas diferentes podem assumir o papel de testador** em diferentes níveis de teste.

Pessoas diferentes podem assumir esses papéis em momentos diferentes. Por exemplo, o papel de gerenciamento de testes pode ser desempenhada por um Líder de Equipe, por um Gerente de Teste, por um Gerente de Desenvolvimento etc.
Também é possível que uma pessoa assuma o papel de testador e gerenciamento de teste ao mesmo tempo.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

1.5 HABILIDADES ESSENCIAIS E BOAS PRÁTICAS EM TESTES

FL-1.5.1 (K2) Dar exemplos das habilidades genéricas necessárias para testar
FL-1.5.2 (K1) Relembrar as vantagens da abordagem de equipe completa
FL-1.5.3 (K2) Distinguir os benefícios e as desvantagens da independência dos testes

1.5.1 HABILIDADES GENÉRICAS NECESSÁRIAS PARA TESTES

Habilidades relevantes para os Testadores:

- Conhecimento sobre testes
- Meticulosidade, cuidado, curiosidade, atenção aos detalhes, ser metódico
- Boas habilidades de comunicação, ser um bom ouvinte, e trabalhar em equipe
- Pensamento analítico, pensamento crítico, criatividade
- Conhecimento técnico
- Conhecimento do domínio

1.5.1 HABILIDADES GENÉRICAS NECESSÁRIAS PARA TESTES

Os Testadores geralmente são os **portadores de más notícias**.

É uma característica humana comum **culpar o portador pelas más notícias**.

Isso torna as **habilidades de comunicação cruciais** para os Testadores.

1.5.1 HABILIDADES GENÉRICAS NECESSÁRIAS PARA TESTES

O **viés de confirmação** pode dificultar o aceite de informações que discordem das crenças atuais.

O teste pode ser considerado uma **atividade destrutiva**, trazendo **crítica contra o produto e o autor**.

Defeitos devem ser **comunicados de forma construtiva**.

1.5.2 ABORDAGEM DE EQUIPE COMPLETA

- É uma **prática do XP** (Extreme Programming)
- **Qualquer membro** da equipe com o conhecimento e as habilidades necessárias **pode executar qualquer tarefa**
- **Todos** são responsáveis pela **qualidade**
- Membros da equipe compartilham o **mesmo espaço de trabalho** (físico ou virtual)
- Essa abordagem **melhora a dinâmica e comunicação** entre a equipe

1.5.2 ABORDAGEM DE EQUIPE COMPLETA

- Testadores **trabalham em estreita colaboração com outros membros da equipe** para garantir que os níveis de qualidade desejados sejam alcançados.
- Isso inclui **colaborar com representantes do negócio** para ajudá-los a criar testes de aceite adequados e **trabalhar com desenvolvedores** para ajudar na estratégia de teste e decidir sobre abordagens de automação de teste.
- Assim, os Testadores podem **transferir conhecimento sobre testes para outros membros da equipe** e influenciar o desenvolvimento do produto.

1.5.2 ABORDAGEM DE EQUIPE COMPLETA

- **Dependendo do contexto, a abordagem de equipe completa nem sempre pode ser adequada.** Por exemplo, em algumas situações, como em sistemas críticos, pode ser necessário um alto nível de independência dos testes.

1.5.3 INDEPENDÊNCIA DOS TESTES

DECO RAR!

- São testes **realizados por alguém que não seja o autor do código**. (Se eu que desenvolvi, outra pessoa tem que testar.
- Existem diferentes **níveis de independência** (quanto mais isolado o teste do desenvolvimento, mais independente)

Sem independência	Alguma independência	Alta independência	Independência muito alta
•**Sem testadores independentes**, desenvolvedores testam seu próprio código.	•Desenvolvedores independentes ou **testadores dentro das equipes de desenvolvimento** ou da equipe do projeto; isso poderiam ser desenvolvedores testando os produtos de seus colegas	•**Realizado por testadores de fora da equipe do autor**	•**Testadores independentes externos à organização**, trabalhando dentro ou fora do escritório.

1.5.3 INDEPENDÊNCIA DOS TESTES

Vantagens

- Encontrar **diferentes tipos de falhas** em comparação a os desenvolvedores
- São mais **imparciais**, pois não possuem premissas de ideias sobre o que testar e não possuem preconceitos ou **apego emocional** com o software.
- Um testador independente pode **verificar, desafiar ou refutar as suposições** feitas pelos stakeholders durante a especificação e a implementação do sistema.

Desvantagens

- **Isolamento** da equipe de desenvolvimento, levando a **uma falta de colaboração**, atrasos no fornecimento de feedback ou um relacionamento adverso.
- Os desenvolvedores podem **perder o senso de responsabilidade pela qualidade**
- Testadores independentes podem ser vistos como **gargalo ou culpados por atrasos** na liberação.
- Testadores independentes podem **não ter algumas informações** importantes (p. ex., sobre o objeto de teste).

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

CAPÍTULO 2

Testes ao longo do Ciclo de Vida de Desenvolvimento de Software

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

2.1 TESTES AO LONGO DO CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE

FL-2.1.1 (K2) Explicar o impacto do ciclo de vida de desenvolvimento de software escolhido nos testes.
FL-2.1.2 (K1) Relembrar as boas práticas de teste que se aplicam a todos os ciclos de vida de desenvolvimento de software.
FL-2.1.3 (K1) Relembrar os exemplos de abordagens de desenvolvimento que priorizam o teste.
FL-2.1.4 (K2) Resumir como DevOps pode ter um impacto nos testes.
FL-2.1.5 (K2) Explicar a abordagem shift-left.
FL-2.1.6 (K2) Explicar como as retrospectivas podem ser usadas como um mecanismo de melhoria de processos.

2.1 TESTES NO CONTEXTO DE UM CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE

Descreve os tipos de atividades realizadas em cada estágio de um projeto de desenvolvimento de software e **como as atividades se relacionam** umas com as outras de forma lógica e cronológica.

Há vários modelos de ciclo de vida de desenvolvimento de software, cada um dos quais requer **abordagens diferentes para o teste**.

Exemplo Modelo Cascata

2.1 TESTES NO CONTEXTO DE UM CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE

Modelos de Ciclo de Vida de Desenvolvimento (SDLC)

- **Sequencial** (ex: modelo em cascata, modelo em V)
- **Iterativo e incrementais** (ex: espiral, modelos ágeis em geral)

Outras práticas:

- XP
- TDD
- ATDD
- BDD
- etc.

MODELOS DE CICLO DE VIDA

Independentemente do modelo de ciclo de vida escolhido, as atividades de teste devem começar nos estágios iniciais, (princípio de testar do início).

2.1.1 IMPACTO DO CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE NOS TESTES

O teste deve ser **adaptado ao SDLC**. A escolha do SDLC tem impacto sobre:

- O **escopo e cronograma** das atividades de teste;
- O **nível de detalhamento** da documentação de teste;
- A **escolha das técnicas** de teste e da abordagem de teste;
- A **extensão da automação** de testes;
- O **papel e responsabilidades** de um Testador;

2.1.1 IMPACTO DO CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE NOS TESTES

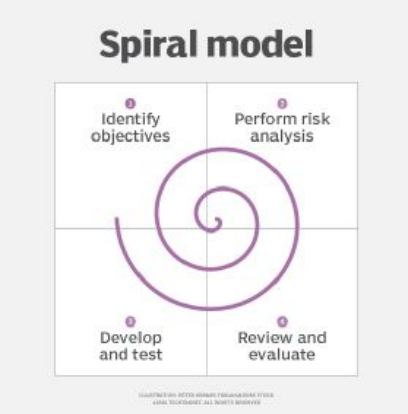
MODELOS SEQUENCIAIS

- Testadores normalmente **participam das revisões de requisitos**, da análise de testes e do projeto de testes.
- Como o código é criado em fases posteriores à criação dos requisitos, **testes dinâmicos não podem ser executados** no início do SDLC.

Exemplo Modelo Cascata

MODELOS ITERATIVOS E INCREMENTAIS

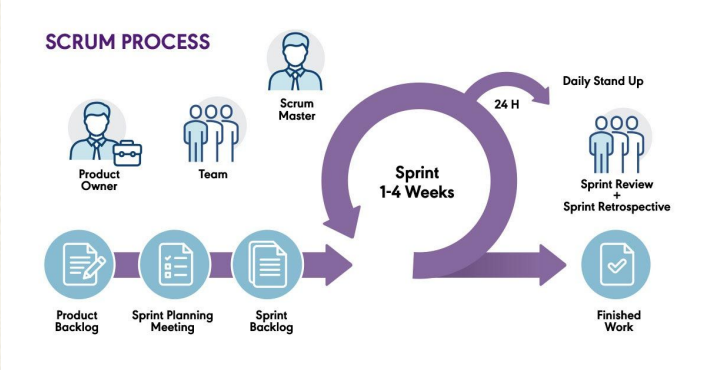
- Em cada iteração, os testes **estáticos e dinâmicos podem ser realizados** em todos os níveis de teste
- **Feedback rápido e testes de regressão extensivos** são necessários devido à entrega frequente de incrementos



2.1.1 IMPACTO DO CICLO DE VIDA DE DESENVOLVIMENTO DE SOFTWARE NOS TESTES

DESENVOLVIMENTO ÁGIL

- **Mudanças podem ocorrer** ao longo do projeto
- Documentação **leve**
- **Ampla automação de testes** para facilitar os testes de regressão
- A maior parte dos **testes manuais tende a ser feita usando técnicas de teste baseadas na experiência** (que não exigem análise e projeto de teste prévio extensivo)



BOAS PRÁTICAS DE TESTE, INDEPENDENTE DO MODELO DE SDLC

- Para **cada atividade de desenvolvimento, há uma atividade de teste** correspondente
- Diferentes **níveis de teste têm objetivos de teste específicos** e diferentes, evitando redundância
- A **análise e a modelagem do teste começam durante a fase de desenvolvimento** correspondente do SDLC, princípio do teste antecipado
- Os **Testadores estão envolvidos na revisão dos produtos de trabalho** assim que os rascunhos dessa documentação estiverem disponíveis, de modo que esse teste antecipado e a detecção de defeitos possam apoiar a estratégia shift-left.

2.1.3 TESTE COMO UM MOTIVADOR PARA O DESENVOLVIMENTO DE SOFTWARE

O TDD, ATDD e BDD são abordagens de desenvolvimento semelhantes, em que os **testes são definidos como um meio de direcionar o desenvolvimento**.

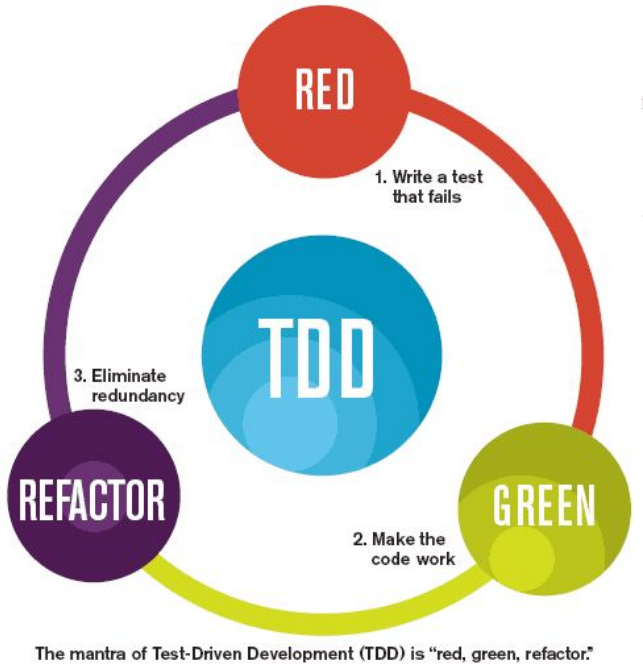
Cada uma dessas abordagens **implementa o princípio do teste antecipado e segue uma abordagem shift-left**, pois os testes são definidos antes de o código ser escrito.

Elas dão suporte a um modelo de desenvolvimento iterativo.

2.1.3 TESTE COMO UM MOTIVADOR PARA O DESENVOLVIMENTO DE SOFTWARE

DESENVOLVIMENTO ORIENTADO POR TESTES (TDD - TEST DRIVEN DEVELOPMENT)

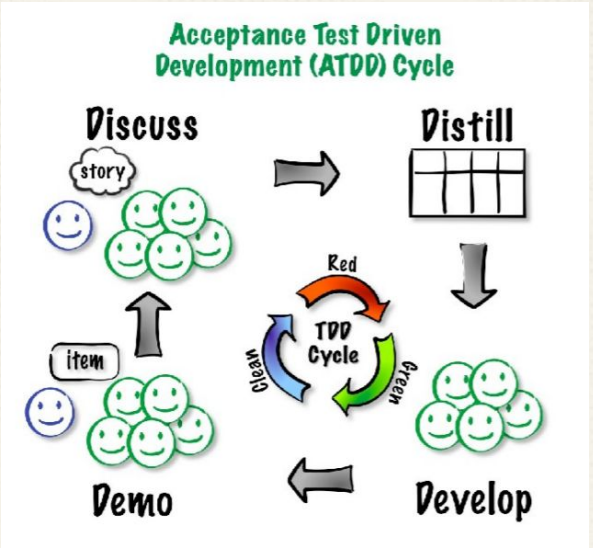
- Direciona a codificação por meio de casos de teste
- Os testes são escritos primeiro, depois o código é escrito para satisfazer os testes e, em seguida, os testes e o código são refatorados



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

DESENVOLVIMENTO ORIENTADO POR TESTE DE ACEITE (ATDD - ACCEPTANCE TEST DRIVEN DEVELOPMENT)

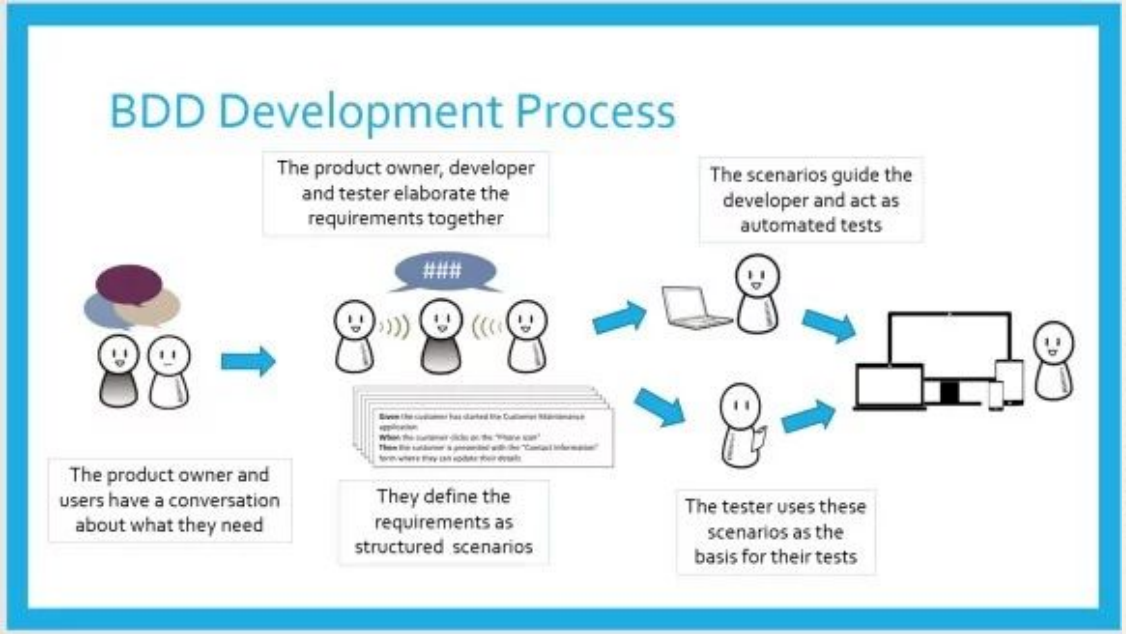
- Deriva testes de critérios de aceite como parte do processo de desenho do sistema
- Os testes são escritos antes que a parte do aplicativo relacionada seja desenvolvida para atender aos testes.



DESENVOLVIMENTO ORIENTADO PELO COMPORTAMENTO (BDD - BEHAVIOR DRIVEN DEVELOPMENT)

- Expressa o comportamento desejado com casos de teste escritos em uma forma simples de linguagem natural, que é fácil de entender pelos stakeholders - geralmente usando o formato Dado/Quando/Então
- Os casos de teste são então traduzidos automaticamente em testes executáveis.

DESENVOLVIMENTO ORIENTADO PELO COMPORTAMENTO (BDD - BEHAVIOR DRIVEN DEVELOPMENT)



DEV - DESENVOLVIMENTO (DEVELOPMENT)

OPS - OPERAÇÕES (OPERATIONS)

- DevOps é uma **abordagem organizacional** que visa a criar sinergia, fazendo com que o desenvolvimento (incluindo os testes) e as operações **trabalhem juntos**

- **Exige mudança cultural na organização** para preencher as lacunas entre o desenvolvimento (incluindo testes) e as operações.

2.1.4 DEVOPS E TESTES

- O DevOps promove:
  - **Autonomia** da equipe
  - **Feedback rápido**
  - **Ferramentas** integradas
  - Práticas técnicas como
    - Integração Contínua (CI - Continuous Integration) e
    - Entrega Contínua (CD - Continuous Delivery)

- Isso permite que as equipes criem, testem e liberem códigos de alta qualidade mais rapidamente por meio de um pipeline de entrega de DevOps

2.1.4 DEVOPS E TESTES

Benefícios do DevOps para Testes:

- Feedback rápido sobre a **qualidade do código** e se as alterações afetam negativamente o código existente;
- O CI promove uma **abordagem shift-left** nos testes, incentivando os desenvolvedores a enviar códigos de alta qualidade acompanhados de testes de componentes e análise estática;
- Promove **processos automatizados**, como CI/CD, que facilitam o estabelecimento de ambientes de teste estáveis;

2.1.4 DEVOPS E TESTES

Benefícios do DevOps para Testes:

- Aumenta a visão das **características de qualidade não funcionais** (ex: performance, confiabilidade);
- A automação por meio de um pipeline de entrega **reduz a necessidade de testes manuais repetitivos**;
- O **risco na regressão é minimizado** devido à escala e ao alcance dos testes de regressão automatizados;

2.1.4 DEVOPS E TESTES

Riscos e Desafios do DevOps:

- O **pipeline de entrega** de DevOps deve ser definido e estabelecido;
- As **ferramentas** de CI/CD devem ser **introduzidas e mantidas**;
- A automação de testes requer recursos adicionais e **pode ser difícil de estabelecer e manter**.

DEVOPS NECESSITA DE UM ALTO NÍVEL DE TESTES AUTOMATIZADOS, PORÉM TESTES MANUAIS AINDA SÃO NECESSÁRIOS (PRINCIPALMENTE DA PERSPECTIVA DO USUÁRIO)

2.1.5 ABORDAGEM SHIFT-LEFT

É UM SINÔNIMO DO PRINCÍPIO DO TESTE ANTECIPADO

LEFT (ESQUERDA), POIS ADICIONA O TESTE NAS ETAPAS INICIAIS (NORMALMENTE ILUSTRADAS À ESQUERDA)

2.1.5 ABORDAGEM SHIFT-LEFT

O shift-left sugere que os **testes devem ser feitos mais cedo** (ex: não esperar que o código seja implementado ou que os componentes sejam integrados)

Mas isso **não significa que os testes posteriores no SDLC devam ser negligenciados**

Testes executados mais cedo vão **otimizar os testes das outras etapas**, pois tendem a encontrar defeitos com antecedência.

2.1.5 ABORDAGEM SHIFT-LEFT

Boas práticas:

- **Revisão da especificação sob a perspectiva de testes.** Essas atividades de revisão das especificações geralmente encontram possíveis defeitos, como ambiguidades, incompletude e inconsistências;
- **Escrever casos de teste antes de o código ser escrito** e fazer com que o código seja executado em um conjunto de testes durante a sua implementação;
- Usar a CI e, melhor ainda, a CD, pois ela vem com **feedback rápido** e testes de componentes automatizados para acompanhar o código-fonte quando ele é enviado ao repositório de código;

2.1.5 ABORDAGEM SHIFT-LEFT

Boas práticas:

- Concluir a **análise estática do código-fonte antes do teste dinâmico** ou como parte de um processo automatizado;
- **Realizar testes não funcionais começando no nível de teste do componente, sempre que possível.** Essa é uma forma de shift-left, pois esses tipos de testes não funcionais tendem a ser realizados mais tarde no SDLC, quando um sistema completo e um ambiente de teste representativo estão disponíveis.

2.1.5 ABORDAGEM SHIFT-LEFT

- Pode ser necessário **treinamento, esforço e/ou custos adicionais** no início do processo, mas espera-se que economize esforços e/ou custos no final do processo.

- Para a abordagem shift-left, é importante que os **stakeholders sejam convencidos a aceitar esse conceito**.

2.1.6 RETROSPECTIVAS E MELHORIAS DE PROCESSO

- Também conhecidas como "reuniões pós-projeto" e retrospectivas de projeto
- **Geralmente são realizadas no final de um projeto ou de uma iteração**, em um marco de lançamento, ou podem ser realizadas quando necessário.
- O momento e a organização das retrospectivas **dependem do modelo específico de SDLC** que está sendo seguido.

2.1.6 RETROSPECTIVAS E MELHORIAS DE PROCESSO

Nessas reuniões, os participantes (não apenas os Testadores, mas também, por exemplo, Desenvolvedores, Arquitetos, Product Owner, Analistas de Negócios) discutem:

- O que foi bem-sucedido e deve ser mantido? (**O que deu certo?**)
- O que não foi bem-sucedido e poderia ser melhorado? (**O que não foi tão legal?**)
- Como incorporar as melhorias e manter os sucessos no futuro? (**Como podemos melhorar?**)

2.1.6 RETROSPECTIVAS E MELHORIAS DE PROCESSO

- Os resultados devem ser registrados e normalmente fazem parte do relatório de conclusão do teste.

- As retrospectivas são essenciais para a implementações bem-sucedidas da melhoria contínua e é importante que todas as melhorias recomendadas sejam acompanhadas.

2.1.6 RETROSPECTIVAS E MELHORIAS DE PROCESSO

Benefícios das retrospectivas para os testes incluem:

- **Aumento da eficácia/eficiência do teste** (ex: implementando sugestões de aprimoramento do processo);
- **Aumento da qualidade do material de teste** (ex: por meio da revisão conjunta dos processos de teste);
- **Vínculo e aprendizado da equipe** (ex: como resultado da oportunidade de levantar questões e propor pontos de melhoria);
- **Melhoria da qualidade da base de teste** (ex: como as deficiências na extensão e na qualidade dos requisitos podem ser abordadas e resolvidas);
- **Melhor cooperação entre desenvolvimento e testes** (ex: como a colaboração é revisada e otimizada regularmente);

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

2.2 NÍVEIS DE TESTE E TIPOS DE TESTE

FL-2.2.1 (K2) Distinguir os diferentes níveis de teste.
FL-2.2.2 (K2) Distinguir os diferentes tipos de teste.
FL-2.2.3 (K2) Distinguir o teste de confirmação do teste de regressão.

NÍVEIS DE TESTE

Grupos de atividades de teste que são organizados e gerenciados juntos
Estão **relacionados a outras atividades** dentro do SDLC
Níveis de teste **podem se sobrepor** no tempo.

TIPOS DE TESTE

São grupos de atividades de teste **relacionadas a características de qualidade específicas** e a maioria dessas atividades de teste **pode ser realizada em todos os níveis de teste**

2.2.1 NÍVEIS DE TESTE

Os **níveis de teste** usados no syllabus são:

- **Teste de componente (unidade)**
- **Teste de Integração entre componentes**
- **Teste de sistema**
- **Teste de Integração entre sistemas**
- **Teste de aceite**

    Aceite de usuário (UAT)

    Aceite operacional

    Aceite contratual e normativo

    Alfa e Beta Teste

2.2.1 NÍVEIS DE TESTE

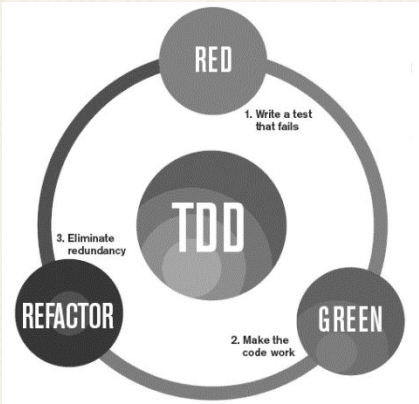
Os níveis de teste são caracterizados pelos seguintes **atributos**:

- Objeto de teste (ou seja, o que está sendo testado);
- Objetivos do teste;
- Base de teste, referenciada para derivar casos de teste;
- Defeitos e falhas;
- Abordagens e responsabilidades específicas

Para cada nível de teste, é necessário um **ambiente de teste adequado**.

TESTE DE COMPONENTE (UNIDADE)

- É o teste na **menor parte testável** de um sistema (Função / Componente / Classe)
- Requer **acesso ao código-fonte**
- Normalmente **realizado pelo próprio desenvolvedor**
- Deve ser **isolado**, **sem integração nenhuma** com outro componente ou parte do sistema
- É um teste muito rápido de ser executado
- No ágil, esse teste pode ser feito antes de desenvolver o código (TDD)
- Deve ser um **teste automatizado**
- Testes de regressão **automatizados** trazem **confiabilidade** nesse nível



TESTE DE COMPONENTE (UNIDADE)

Objetivos:

- Reduzir o risco;
- Verificar os comportamentos funcional e não funcional do componente ao projetado e especificado;
- Construir a confiança na qualidade do componente;
- Encontrar defeitos no componente;
- **Evitar que os defeitos espalhem para níveis mais altos de teste**

TESTE DE INTEGRAÇÃO

- **Foca na integração** entre Componentes ou Sistemas.
- Requer **acesso ao código-fonte** (para integração entre componentes)
- Pode ser **realizado pelo desenvolvedor** (componentes) **ou testador** (sistemas)
- Quanto **maior o escopo** da integração, **mais difícil é isolar os defeitos**, por isso a **integração contínua** é comum hoje em dia.
- Testes de regressão **automatizados** trazem **confiabilidade** nesse nível

TESTE DE INTEGRAÇÃO

Os **objetivos do teste de integração**:

- Reduzir risco;
- Verificar se os comportamentos funcionais e não funcionais das interfaces estão projetados e especificados;
- Construir confiança na qualidade das interfaces;
- Encontrar defeitos (que podem estar nas próprias interfaces ou nos componentes ou sistemas);
- Evitar que os defeitos espalhem para níveis mais altos de teste.

TESTE DE INTEGRAÇÃO

- O teste de **INTEGRAÇÃO DE COMPONENTES** foca nas interações e interfaces entre **componentes integrados**. É executado após o teste do componente e **geralmente é automatizado**. No desenvolvimento iterativo e incremental, os testes de integração de componentes **geralmente fazem parte do processo de integração contínua**;

- O teste de **INTEGRAÇÃO DE SISTEMAS** concentra-se nas interações e **interfaces entre sistemas**, pacotes e microserviços. O teste de integração do sistema também pode abranger interações e interfaces fornecidas por **entidades externas** (p. ex., serviços da Web).

TESTE DE SISTEMA

- Teste do Sistema em si sendo **executado**, próximo ao cenário do usuário final
- Fluxos de **ponta a ponta** (end-to-end)
- Produz **informações** que são usadas para **tomar decisões** de liberação
- Também pode verificar **requisitos legais** / regulatórios
- Testes de regressão **automatizados** trazem **confiabilidade** nesse nível
- Executado pelos testadores

TESTE DE SISTEMA

Objetivos:

- Reduzir o risco;
- Verificar se os comportamentos funcionais e não funcionais do sistema estão como projetados e especificados;
- Validar se o sistema está completo e funcionará como esperado;
- Criar confiança na qualidade do sistema como um todo;
- Encontrar defeitos;
- Evitar que os defeitos espalhem para níveis mais altos de teste ou produção.

TESTE DE ACEITE

Concentra-se na validação e na demonstração do sistema.
Preferencialmente, o teste de aceite deve ser realizado pelos usuários previstos.

Não tem como objetivo encontrar defeitos mas sim **garantir que o sistema esteja funcionando da maneira como especificado** para o usuário final.

Defeitos encontrados durante esse teste são considerados **grandes riscos** para o projeto.

Responsáveis: clientes, usuários de negócios, proprietários de produtos, operadores de um sistema e stakeholders.

TESTE DE ACEITE

Teste de Aceite de Usuário (UAT)

- Certificar que o sistema atenda as necessidades do usuário
- Validar o uso do sistema por usuários em um ambiente de produção ou simulado.

Teste de Aceite Operacional (OAT)

- Garantir que os operadores ou administradores do sistema possam manter o sistema funcionando adequadamente para os usuários no ambiente de produção

- São realizados testes como:
  Backups e Restauração
  Performance
  Vulnerabilidades de Seguraça

Teste de Aceite Contratual e Regulatório

- Garantir que a conformidade contratual ou regulatória foi alcançada.
- Critérios devem ser definidos quando as partes assinam o contrato para o desenvolvimento do Software
- Resultados podem ser acompanhados por testemunhas ou agências reguladoras

TESTE DE ACEITE

Teste Alfa e Beta

Alfa

- *Realizado no site (local) da organização.*
- *Pode ser feito por clientes, operadores ou testadores independentes*

Beta

- *Testes realizados fora da organização, ou seja em local próprio.*
- *Pode ser feito por clientes em potencial ou operadores*

TESTE FUNCIONAL

- **“O que”** o sistema deve fazer
- Avalia as **funções** que o sistema deve executar
- Devem ser realizados em **todos os níveis de teste**
- Eficácia medida através da **cobertura funcional**
- Pode exigir conhecimentos especiais na **regra de negócio**

TESTE NÃO FUNCIONAL

- “**Quão bem**” o sistema se comporta
- Avalia as **características** de um software (ex: usabilidade, desempenho, segurança)
- Podem ser realizados em **todos os níveis de teste**
- Eficácia medida através da **cobertura não funcional**
- Pode exigir conhecimentos especiais na **tecnologia** (para vulnerabilidades, desempenho, etc.)

TESTE DE CAIXA PRETA

Baseado nas **especificações** do sistema

Deriva testes da documentação externa ao objeto de teste

O principal objetivo é **verificar o comportamento** do sistema **em relação às suas especificações**

TESTE DE CAIXA BRANCA

Baseado na **estrutura interna** do sistema (código, arquitetura, etc)

Eficácia medida através da **cobertura estrutural** (código, interfaces entre componentes)

Pode exigir conhecimentos especiais em como o **código** é construído

2.2.3 TESTE DE CONFIRMAÇÃO E REGRESSÃO

TESTE DE CONFIRMAÇÃO		TESTE DE REGRESSÃO	
Confirma se um defeito foi corrigido com sucesso	Retestar testes que falharam devido a um defeito	**Garante que o que funcionava, continua funcionando**	Detecta efeitos colaterais

- executar todos os casos de teste que falharam anteriormente devido ao defeito
- adicionar novos testes para cobrir quaisquer alterações necessárias para corrigir o defeito
- quando há pouco tempo ou dinheiro para corrigir defeitos, pode se restringir a simplesmente executar as etapas que devem reproduzir a falha

- conjuntos de testes de regressão são executados muitas vezes
- o número de casos de teste de regressão aumentará a cada iteração ou versão
- testes de regressão são fortes candidatos à automação (incluindo no CI / DevOps)

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

2.3 TESTE DE MANUTENÇÃO

FL-2.3.1 (K2) Resumir os testes de manutenção e seus acionadores.

2.3 TESTE DE MANUTENÇÃO

Depois de implantados em ambientes de produção, o software e os sistemas **precisam ser mantidos**.

- Para **corrigir defeitos**
- Para **adicionar novas funcionalidades**
- Para **remover ou alterar** funcionalidades já entregues.
- Para manter a **eficiência de performance**, segurança, confiabilidade, etc.

2.3 TESTE DE MANUTENÇÃO

- Há diferentes categorias de manutenção, que podem ser **corretivas, adaptáveis a mudanças no ambiente ou melhorar a performance** ou a capacidade de manutenção

- A análise de impacto pode ser feita antes de uma alteração, para ajudar a decidir se a alteração deve ser feita, com base nas possíveis consequências em outras áreas do sistema

2.3 TESTE DE MANUTENÇÃO

- O teste das alterações em um sistema em produção inclui a **avaliação do sucesso da implementação da alteração e a verificação de possíveis regressões** em partes do sistema que permanecem inalteradas (que geralmente é a sua maior parte).

2.3 TESTE DE MANUTENÇÃO

Quando qualquer alteração é feita como manutenção, testes de manutenção devem ser feitos:

- Para avaliar o sucesso com que as **alterações foram feitas**
- Para verificar possíveis **efeitos colaterais** (teste de regressão)

O **escopo** do teste de manutenção depende do:

- Grau de **risco** da mudança
- Tamanho do sistema

2.3 TESTE DE MANUTENÇÃO

Modificação

- Atualizações, correções, melhorias, atualizações operacionais

Atualização / Migração

- Migração ou atualizações de plataforma

Aposentadoria

- Quando um aplicativo atinge o fim de sua vida útil (arquivamento / retenção / restauração dos dados)

DECORAR!

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

CAPÍTULO 3

Teste Estático

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho (@leocarvalho.eu)

3.1 NOÇÕES BÁSICAS DE TESTE ESTÁTICO

FL-3.1.1 (K1) Reconhecer os tipos de produtos que podem ser examinados pelas diferentes técnicas de teste estático.
FL-3.1.2 (K2) Explicar o valor dos testes estáticos.
FL-3.1.3 (K2) Comparar e contrastar testes estáticos e dinâmicos.

3.1 NOÇÕES BÁSICAS DE TESTE ESTÁTICO

REVISÕES

- Exame manual dos produtos de trabalho

ANÁLISE ESTÁTICA

- Avaliação por meio de ferramentas (SonarQube, verificação de ortografia/gramática, etc...)

A avaliação é feita **sem executar** o código

3.1 NOÇÕES BÁSICAS DE TESTE ESTÁTICO

- Os objetivos incluem a **melhoria da qualidade, a detecção de defeitos e a avaliação de características** como legibilidade, integridade, correção, testabilidade e consistência.
- Pode ser aplicado tanto para **verificação quanto para validação**.
- Técnicas de revisão podem ser aplicadas para **garantir que as histórias de usuários sejam completas e compreensíveis e incluam critérios de aceite testáveis**.

3.1 NOÇÕES BÁSICAS DE TESTE ESTÁTICO

- Ao fazer as perguntas certas, os Testadores exploram, desafiam e ajudam a melhorar as histórias de usuário propostas.
- A análise estática **pode identificar problemas antes dos testes dinâmicos** e, muitas vezes, exige **menos esforço**, já que não são necessários casos de teste e, normalmente, são usadas ferramentas.
- A **análise estática** é frequentemente **incorporada às estruturas de CI**.
- Verificadores ortográficos e ferramentas de legibilidade são outros exemplos de ferramentas de análise estática.

3.1.1 PRODUTOS DE TRABALHO EXAMINÁVEIS POR TESTES ESTÁTICOS

Qualquer produto de trabalho pode ser examinado por testes estáticos (revisões ou análise estática):

- Especificações
- Histórias de usuário
- Arquitetura
- Código
- Páginas Web
- Documentação do projeto
- Contratos
- Planos de projeto
- Planos de teste
- Casos de teste
- Cartas de teste
- Etc...

Qualquer produto de trabalho que seja possível ler e entender

3.1.2 VALOR DO TESTE ESTÁTICO

- Pode **detectar defeitos nas primeiras fases do SDLC**, cumprindo o princípio do teste antecipado
- Pode identificar defeitos que **não podem ser detectados por testes dinâmicos**
- Permitem **avaliar a qualidade e criar confiança** nos produtos de trabalho
- Stakeholders também podem se certificar de que **requisitos descrevem suas necessidades reais**
- A **comunicação entre os stakeholders será aprimorada**. Por esse motivo, é recomendável envolver uma grande variedade de participantes nos testes estáticos.
- Por mais que exija tempo e investimento, os **custos gerais do projeto geralmente são muito menores** do que quando não são realizadas revisões, pois diminui a incidência de defeitos em etapas posteriores
- Os **defeitos de código podem ser detectados usando a análise estática de forma mais eficiente** do que em testes dinâmicos

3.1.3 DIFERENÇAS ENTRE TESTE ESTÁTICO E DINÂMICO

- Se complementam, encontrando **diferentes tipos de defeitos**
- **Estático encontra defeitos direto nos produtos de trabalho** (e não falhas, quando o software está em execução).
- **Estático** pode ser usado para melhorar a **qualidade interna** dos produtos de trabalho, já o **dinâmico** se concentra em **comportamentos externamente visíveis**.
- Defeitos encontrados no Estático são **mais baratos de corrigir**

3.1.3 DIFERENÇAS ENTRE TESTE ESTÁTICO E DINÂMICO

Em comparação com o teste dinâmico, os **defeitos mais comuns que são mais fáceis e baratos** de encontrar e corrigir por meio de teste estático inclui:

- Defeitos em requisitos (inconsistências, ambiguidades, contradições, omissões, imprecisões e redundâncias);
- Defeitos de desenho (algoritmos ineficientes ou estruturas de banco de dados, alto acoplamento, baixa coesão);
- Defeitos de codificação (variáveis com valores indefinidos, variáveis declaradas e nunca utilizadas, código inacessível, código duplicado);
- Desvios dos padrões (falta de aderência aos padrões de codificação);
- Especificações de interface incorretas (unidades de medida diferentes usadas pelo sistema de chamada do que pelo sistema chamado);
- Vulnerabilidades de segurança;

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

3.2 PROCESSO DE FEEDBACK E REVISÃO

FL-3.2.1 (K1) Identificar os benefícios do feedback antecipado e frequente dos stakeholders.
FL-3.2.2 (K2) Resumir as atividades do processo de revisão.
FL-3.2.3 (K1) Relembrar quais responsabilidades são atribuídas às funções principais ao realizar revisões.
FL-3.2.4 (K2) Comparar e contrastar os diferentes tipos de revisão.
FL-3.2.5 (K1) Relembrar os fatores que contribuem para uma revisão bem-sucedida.

3.2.1 BENEFÍCIOS DO FEEDBACK ANTECIPADO E FREQUENTE DOS STAKEHOLDERS

- Permite a **comunicação precoce de possíveis problemas** de qualidade.
- Pode **evitar mal-entendidos sobre os requisitos** e garantir que as alterações nos requisitos sejam compreendidas e implementadas mais cedo.
- **Melhorar a compreensão** do que está sendo desenvolvido.
- **Permite que a equipe se concentre nos recursos que agregam mais valor** aos stakeholders e que têm o maior impacto positivo sobre os riscos identificados.

3.2.2 ATIVIDADES DO PROCESSO DE REVISÃO

PLANEJAMENTO	INÍCIO DA REVISÃO	REVISÃO INDIVIDUAL	COMUNICAÇÃO E ANÁLISE DE PROBLEMAS	CORREÇÃO E RELATÓRIO
Definição de: - Escopo da revisão - Objetivo - Produto de trabalho a ser revisado - Caract. De qualidade a serem avaliadas - Critérios de saída - Padrões / Esforço / Prazo	- Garantir que todos estejam preparados para a revisão. - Garantir que todos tenham acesso ao produto de trabalho - Garantir que todos entendam suas funções e responsabilidades e recebam tudo o que for necessário para realizar a análise.	- Cada revisor realiza uma revisão individual para avaliar a qualidade do produto de trabalho - Observar possíveis defeitos, recomendações e questões	- Comunicar anomalias encontradas - Analisar anomalias encontradas (nem todas podem ser defeitos) - Tomar decisão sobre status e ações para cada anomalia	- Deve ser criado um relatório de defeitos para que as ações corretivas possam ser acompanhadas. - Quando os critérios de saída forem atingidos, o produto de trabalho poderá ser aceito. - Os resultados da revisão são relatados.

By Leonardo Carvalho

3.2.3 FUNÇÕES E RESPONSABILIDADES EM UMA REVISÃO FORMAL

GERENTE	AUTOR	MODERADOR (OU FACILITADOR)	RELATOR (OU REGISTRADOR)	REVISOR	LÍDER DA REVISÃO
Decide o que deve ser revisado e fornece recursos, como equipe e tempo para a revisão	Cria e corrige o produto de trabalho em análise	Garante o andamento eficaz das reuniões de revisão, incluindo mediação, gerenciamento de tempo e um ambiente de revisão seguro no qual todos possam falar livremente	Reúne as anomalias dos revisores e registra as informações da revisão	Realiza revisões. Pode ser alguém que esteja trabalhando no projeto, um especialista no assunto ou qualquer outra parte interessada	Assume a responsabilidade geral pela revisão, como decidir quem estará envolvido e organizar quando e onde a revisão será realizada

Uma pessoa pode desempenhar mais de uma função.

By Leonardo Carvalho

3.2.4 TIPOS DE REVISÃO

DECORAR!

INFORMAL	ACOMPANHAMENTO (walkthrough)	REVISÃO TÉCNICA	INSPEÇÃO
• Sem processo formal • Objetivo detectar anomalias e resolver rapidamente pequenos problemas • Pode ser realizado por um colega ou mais pessoas • Documentar resultados é opcional • Mais usado no Ágil	• Conduzidas pelo próprio autor • Pode servir a muitos objetivos, como avaliar a qualidade, criar confiança no produto do trabalho, instruir os revisores, obter consenso, gerar novas ideias, motivar e permitir que os autores melhorem e detectar anomalias. • Os revisores podem realizar uma revisão individual antes, mas isso não é obrigatório.	• É realizada por revisores tecnicamente qualificados e liderada por um moderador • Objetivo obtenção de consenso sobre problemas técnicos e detecção de anomalias, avaliar a qualidade e criar confiança	• São o tipo mais formal de revisão • O objetivo principal é encontrar o número máximo de anomalias. • Outros objetivos são avaliar a qualidade, criar confiança no produto do trabalho e motivar e permitir que os autores melhorem. •As métricas são coletadas e usadas para aprimorar o SDLC. •Nas inspeções, o autor não pode atuar como líder ou relator da revisão.

3.2.5 FATORES DE SUCESSO PARA REVISÕES

- Definir objetivos claros e critérios de saída mensuráveis.
- Escolher o tipo de revisão apropriado para atingir os objetivos e se adequar ao tipo de produto de trabalho, aos participantes da revisão, às necessidades e ao contexto do projeto.
- Realizar revisões em pequenas partes, de modo que os revisores não percam a concentração.
- Fornecer feedback das revisões aos stakeholders e aos autores para que eles possam melhorar o produto e suas atividades.
- Fornecer tempo suficiente para que os participantes se preparem para a revisão;
- Apoio da gerência para o processo de revisão;
- Tornar as revisões parte da cultura da organização, para promover o aprendizado e o aprimoramento dos processos.
- Fornecer treinamento adequado a todos os participantes sobre suas funções.
- Ter facilitação de reuniões.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

CAPÍTULO 4

Análise e Modelagem de Teste

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

4.1 VISÃO GERAL DAS TÉCNICAS DE TESTE

FL-4.1.1 (K2) Distinguir técnicas de teste baseadas em caixa-preta, caixa-branca e experiência.

4.1 VISÃO GERAL DAS TÉCNICAS DE TESTE

- Dão **suporte ao Testador na análise** do teste (o que testar) e no projeto do teste (como testar).
- Ajudam a **desenvolver um conjunto relativamente pequeno, mas suficiente**, de casos de teste de forma sistemática.
- Também **ajudam o Testador a definir as condições de teste**, identificar os itens de cobertura e identificar os dados de teste durante a análise e o projeto do teste.

4.1 VISÃO GERAL DAS TÉCNICAS DE TESTE

CAIXA-PRETA (ou baseadas em especificações)

- São baseadas em uma análise do comportamento específico do objeto de teste **sem referência à sua estrutura interna**.
- Os casos de teste são independentes de como o software é implementado.
- Se a implementação mudar, mas o comportamento exigido permanecer o mesmo, os casos de teste ainda serão úteis.

CAIXA-BRANCA (ou baseadas na estrutura)

- Baseiam-se em uma análise da estrutura interna e do processamento do objeto de teste.
- Como os casos de teste dependem de como o software é projetado, eles só podem ser criados após o projeto ou a implementação do objeto de teste.

BASEADAS NA EXPERIÊNCIA

- Usam efetivamente o conhecimento e a experiência dos Testadores para o projeto e a implementação dos casos de teste.
- A eficácia dessas técnicas depende muito das habilidades do Testador.
- Podem detectar defeitos que podem não ser detectados usando as técnicas de teste caixa-preta e de caixa branca.
- São complementares às técnicas de teste caixa-preta e de caixa-branca.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

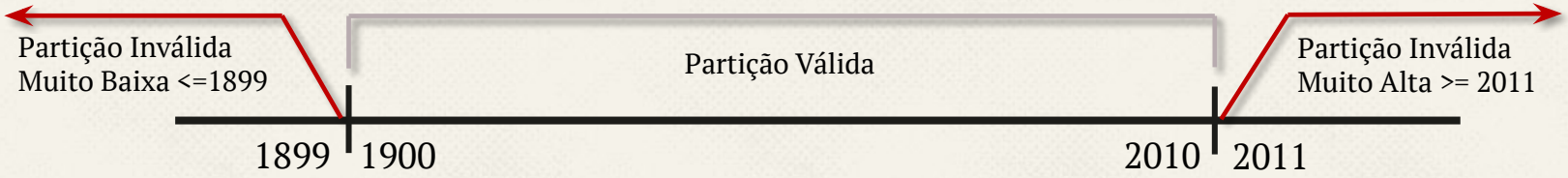
4.2 TÉCNICAS DE TESTE DE CAIXA-PRETA

FL-4.2.1 (K3) Usar o particionamento de equivalência para derivar casos de teste.
FL-4.2.2 (K3) Usar a análise de valor limite para derivar casos de teste.
FL-4.2.3 (K3) Usar testes de tabela de decisão para derivar casos de teste.
FL-4.2.4 (K3) Usar o teste de transição de estado para derivar casos de teste.

4.2.1 PARTICIONAMENTO DE EQUIVALÊNCIA (EC)

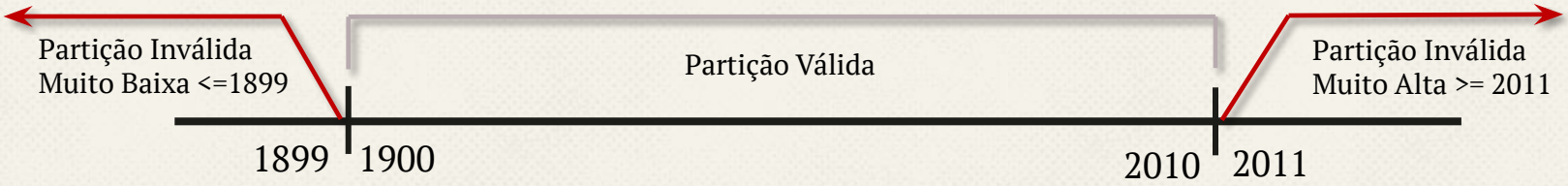
- Divide os dados em **Partições** (ou Classes de Equivâlência)
- Todos os membros de uma partição **devem ser processados da mesma maneira** (são equivalentes)
- Partição **Válida** contém **valores** que devem ser **aceitos**
- Partição **Inválida** contém **valores** que devem ser **rejeitados**
- Cada valor deve pertencer a **apenas uma** partição de equivalência
- **Cobertura** = Partições testadas / Total de partições

Imagine um campo de ano de nascimento que aceita valores de 1900 até 2010



4.2.1 PARTICIONAMENTO DE EQUIVALÊNCIA (EC)

Imagine um campo de ano de nascimento que aceita valores de 1900 até 2010



Qual é o mínimo de casos de teste necessários para cobrir todas as partições?

3, sendo:

- **1 teste com qualquer valor <=1899**
- **1 teste com qualquer valor entre 1900 e 2010**
- **1 teste com qualquer valor >=2011**

By Leonardo Carvalho

4.2.1 PARTICIONAMENTO DE EQUIVALÊNCIA (EC)

Questão 22

Um gravador de radiação diário para plantas produz uma pontuação de luz solar baseada na combinação do número de horas que uma planta está exposta ao sol (abaixo de 3 horas, 3 a 6 horas ou acima de 6 horas) e a intensidade média da luz solar (muito baixa, baixa, média, alta).

Dados os seguintes casos de teste:

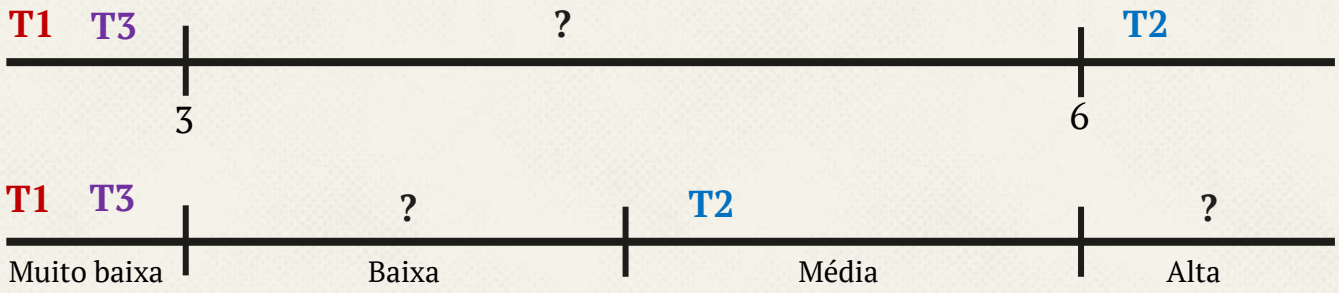
	Horas	Intensidade	Placar
T1	1,5	Muito baixa	10
T2	7,0	Média	60
T3	0,5	Muito baixa	10

Qual é o número mínimo de casos de teste adicionais que são necessários para garantir a cobertura total de TODAS as partições de equivalência de entrada válida?

A) 1
B) 2
C) 3
D) 4

Com 2 **Casos de Teste** seria possível cobrir as partições restantes:

- 1 Caso de Teste usando **horas entre 3 e 6** e Intensidade **Baixa**.
- 1 Caso de Teste usando qualquer hora e Intensidade **Alta**



4.2.2 ANÁLISE DE VALOR LIMITE (BVA)

- É uma **extensão** do Particionamento de Equivalência
- Só pode ser usada quando a partição é **ordenada**, consistindo em dados numéricos ou sequenciais
- Os valores **mínimo e máximo** de uma partição **são seus valores limites**
- Pode ser aplicada em **todos os níveis de teste**
- **Maior chance de encontrar bugs do que Partição de Equivalência**
- **Cobertura** = Limites testados / Total de limites

Imagine um campo de ano de nascimento que aceita valores de 1900 até 2010

4.2.2 ANÁLISE DE VALOR LIMITE (BVA)

Imagine um campo de ano de nascimento que aceita valores de 1900 até 2010

Qual é o mínimo de casos de teste necessários para cobrir todos os limites?

4, sendo:

- **1 teste com o valor 1899**
- **1 teste com o valor 1900**
- **1 teste com o valor 2010**
- **1 teste com o valor 2011**

4.2.2 ANÁLISE DE VALOR LIMITE (BVA)

By Leonardo Carvalho

Questão 23

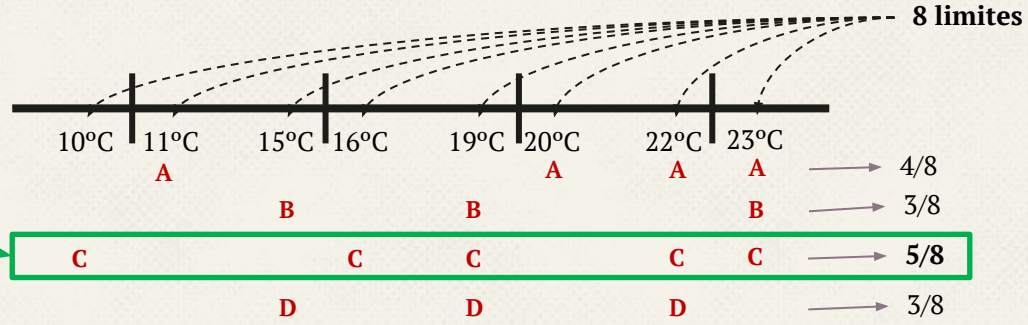
Um aplicativo doméstico inteligente mede a temperatura média na casa durante a semana anterior e fornece feedback aos ocupantes sobre sua compatibilidade ambiental com base nessa temperatura.

O feedback para diferentes faixas de temperatura média (para a temperatura mais próxima) deve ser:

Até 10°C:	Gelado
de 11°C a 15°C:	Refrigerado
16°C a 19°C:	Agradável
20°C a 22°C:	Quente
Acima de 22°C:	Muito quente

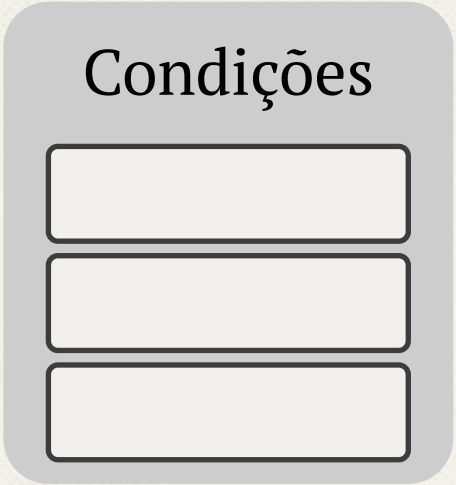
Usando BVA (somente valores Min e Max), qual dos seguintes conjuntos de entradas de teste fornece o mais alto nível de cobertura de limite?

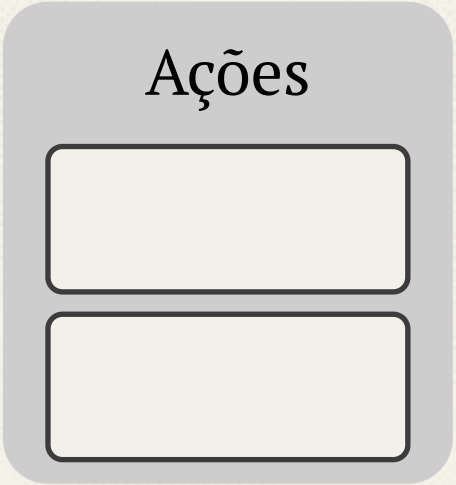
A) 0°C, 11°C, 20°C, 22°C, 23°C
B) 9°C, 15°C, 19°C, 23°C, 100°C
C) 10°C, 16°C, 19°C, 22°C, 23°C
D) 14°C, 15°C, 18°C, 19°C, 19°C, 21°C, 22°C



4.2.3 TABELA DE DECISÃO

- Testam **combinações** que levam à resultados diferentes
- Uma boa maneira de registrar **regras de negócio complexas** que um sistema deve implementar
- É formada de **Condições** (entradas) e **Ações** Resultantes (saídas), que formam as **linhas** da tabela
- **Regras** são as **colunas**, indicando quais condições resultam em quais ações
- Número de Colunas de **Regras** representam o **número de Casos de Teste** (cobertura)





CONDIÇÕES	Regras 1	Regras 2	Regras 3	Regras 4	Regras 5	Regras 6	Regra 7	Regra 8
Ocupa cargo de chefia ?	S	S	S	S	N	N	N	N
Idade maior que 40 anos ?	S	S	N	N	S	S	N	N
Mais de 2 anos no cargo ?	S	N	S	N	S	N	S	N
AÇÕES								
Exame especial	X	X	X		X	X		
Exame normal				X			X	X

onde: S=sim; N=não; X=ação a ser executada.

4.2.3 TABELA DE DECISÃO

Uma empresa decidiu que se as compras forem realizadas até as 23:59 do dia atual, os clientes terão 15% de desconto e se a compra for maior de R$100,00 o frete será grátis. Quantas regras deverão existir e serem analisadas para cobrirem todas as combinações nesse caso?

Condições	Regra 1	Regra 2	Regra 3	Regra 4
<= 23h59	F	V	F	V
>= R$100,00	F	F	V	V
Ações				
Desc. 15%		X		X
Frete Grátis			X	X

4.2.3 TABELA DE DECISÃO

Questão 27

Os funcionários de uma empresa recebem bônus se trabalharem mais de um ano na empresa e atingirem uma meta que tenha sido acordada individualmente antes.

Estes fatos podem ser mostrados em uma tabela de decisão:

Test-ID		T1	T2	T3	T4
Condição 1	Emprego por mais de 1 ano?	YES	NO	NO	YES
Condição 2	Objetivo acordado?	NO	NO	YES	YES
Condição 3	Alcançado o objetivo?	NO	NO	YES	YES
Ação	Pagamento de bônus	NO	NO	NO	YES

Qual dos seguintes casos de teste representa uma situação que pode acontecer na vida real, e está faltando na tabela de decisão acima?

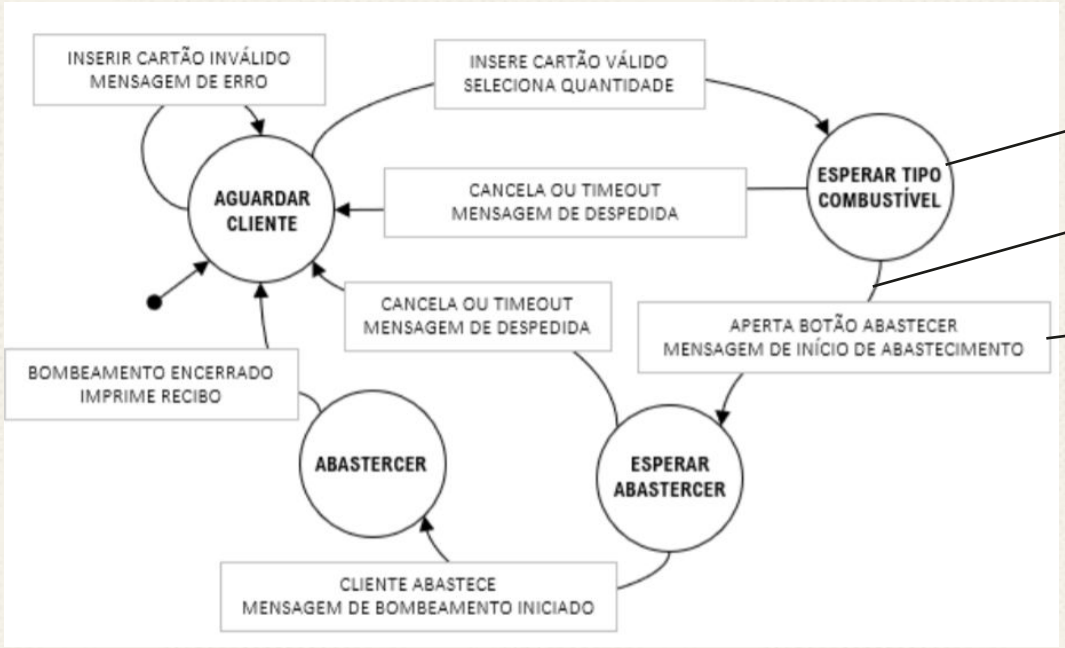
A) Condição1 = SIM, Condição2 = NÃO, Condição3 = SIM, Ação = NÃO
B) Condição1 = SIM, Condição2 = SIM, Condição3 = NÃO, Ação = SIM
C) Condição1 = NÃO, Condição2 = NÃO, Condição3 = SIM, Ação= NÃO
D) Condição1 = NÃO, Condição2 = SIM, Condição3 = NÃO, Ação= NÃO

D) CORRETO. O caso teste descreve a situação de que o período muito curto de emprego e o não cumprimento da meta acordada leva ao não pagamento do bônus.
Esta situação pode ocorrer na prática, mas está ausente na tabela de decisão

4.2.4 TESTE DE TRANSIÇÃO DE ESTADO

- Um diagrama de transição de estado mostra os **possíveis estados** do software, bem como a forma como o software entra, sai e **transita** entre os estados
- Uma transição é iniciada por um evento (p. ex., entrada do usuário de um valor em um campo)
- A **mudança de estado** pode fazer com que o software execute uma **ação** (p. ex., gerar uma mensagem de cálculo ou de erro).
- Uma **tabela de transição de estado** mostra todas as transições **válidas** e potencialmente **inválidas** entre **estados**, bem como os eventos, condições de proteção e ações resultantes para transições válidas.
- Os **diagramas de transição de estado** normalmente mostram **apenas as transições válidas** e excluem as transições inválidas
- Os testes podem ser projetados para **cobrir uma sequência típica de estados**, para exercitar **todos os estados**, para **exercitar cada transição**, para executar sequências específicas de transições ou para testar transições inválidas (cobertura)

4.2.4 TESTE DE TRANSIÇÃO DE ESTADO



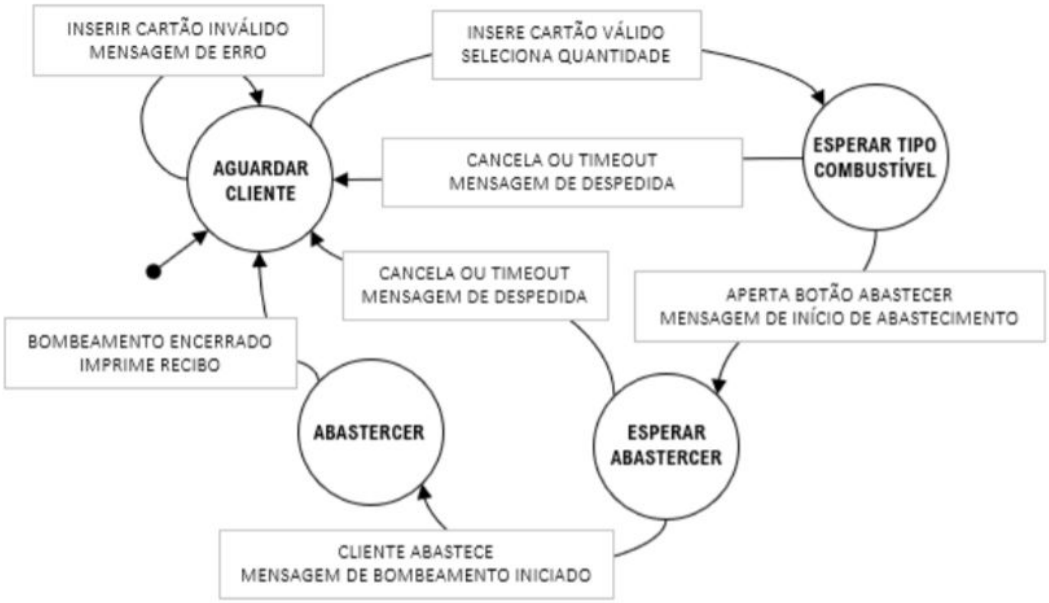
4.2.4 TESTE DE TRANSIÇÃO DE ESTADO

Questão 28

Considere o seguinte diagrama de transição de estado apenas para uma bomba de gasolina sem supervisão, com cartão de crédito:

Suponha que você queira desenvolver o número mínimo de testes para cobrir cada transição no diagrama de transição de estado. Suponha ainda que cada teste deve começar no estado inicial, esperando cliente, e cada teste termina quando uma transição chega ao estado inicial. Quantos testes você precisa?

A) 4
B) 7
C) 1
D) Infinito



CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

4.3 TÉCNICAS DE TESTE DE CAIXA-BRANCA

FL-4.3.1 (K2) Explicar o teste de instrução.
FL-4.3.2 (K2) Explicar o teste de ramificação.
FL-4.3.3 (K2) Explicar o valor dos testes caixa-branca.

4.3 TÉCNICAS DE TESTE DE CAIXA-BRANCA

- Baseado na **estrutura interna** do objeto de teste (classes, funções, módulos...)
- Podem ser usadas em **todos os níveis de teste**, mas as técnicas que estudaremos agora são **mais comuns de serem usadas no teste de componente ou integração entre componentes**
- Medição via **cobertura do teste**
- Geralmente feito **pelo Desenvolvedor**

4.3.1 TESTE E COBERTURA DE INSTRUÇÃO

Instruções (comando ou sentença) são linhas de código que não necessitam de condições para serem executadas.

Essa técnica tem como objetivo testar as **instruções executáveis do código**.

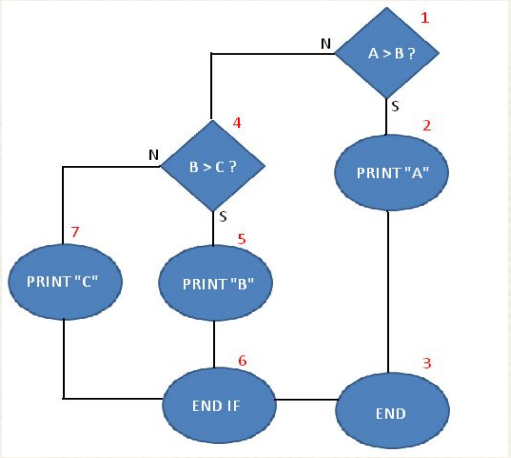
- O **percentual de cobertura** de instrução é medido através da fórmula:
  (Instruções Executadas / Total de Instruções Existentes) * 100

É a técnica de Caixa-Branca **menos eficiente**, pois garante apenas a cobertura das instruções, que **não englobam nenhum outro tipo de cobertura**

A palavra INSTRUÇÃO, também pode ser encontrada como DECLARAÇÃO, COMANDO OU SENTENÇA

4.3.1 TESTE E COBERTURA DE INSTRUÇÃO

```
if (a > b)
    Print (“A”);
else
    if (B > C)
        Print (“B”);
    else
        Print (“C”);
    end if;
end;
```



Se (A=1), (B=2) e (C=3), qual a cobertura de instruções obtida com esse teste (em %)?

Cobertura de Instrução = (5 / 7)*100 = 71% de cobertura com esse teste

4.3.2 TESTE E COBERTURA DE RAMIFICAÇÃO

- Testa as **decisões existentes no código** e o código executado com base nos resultados da decisão

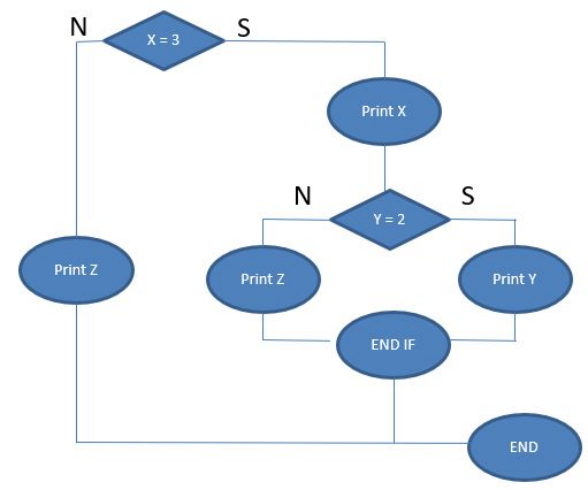
- Casos de Teste criados de para **cobrir as decisões existentes** (para uma instrução IF, um para o resultado verdadeiro e outro para o resultado falso; para uma instrução CASE, os casos de teste seriam necessários para todos os possíveis resultados, incluindo o resultado padrão)

- O **percentual de cobertura de decisão** é medido através da seguinte fórmula:
  (Número de Resultados de Decisão Executados / Total de Decisões Existentes) *100

A palavra DECISÃO, também pode ser encontrada como DESVIO OU RAMIFICAÇÃO

4.3.2 TESTE E COBERTURA DE RAMIFICAÇÃO

```
If x = 3 Then
    Print X
    if y = 2 Then
        Print Y
    Else
        Print Z
Else
    Print Z
End
```



Tendo como base o pseudocódigo acima, quantos testes (ou casos de teste) são necessários para atingir 100% da cobertura de desvio/decisão?

3 Casos de Teste

- **X = 3 e Y = 2**
- **X = 0 e Y = 0**
- **X = 3 e Y = 0**

4.3.2 TESTE E COBERTURA DE RAMIFICAÇÃO

- 100% de cobertura de ramificação garante 100% de cobertura de instrução

4.3.3 O VALOR DO TESTE DE CAIXA BRANCA

- **Toda a implementação do software é levada em conta durante o teste**, o que facilita a detecção de defeitos mesmo quando a especificação do software é vaga, desatualizada ou incompleta.
- Um ponto fraco é que, se o software não implementar um ou mais requisitos, o teste caixa-branca pode não detectar os defeitos resultantes ausentes.
- As técnicas de caixa-branca **podem ser usadas em testes estáticos**.
- São adequadas para **revisar códigos que ainda não estão prontos para execução**.
- Realizar somente testes de **caixa-preta não fornece uma medida da cobertura real do código.**
- As medidas de **cobertura de caixa-branca fornecem uma medição objetiva** da cobertura e fornecem as informações necessárias para permitir que testes adicionais sejam gerados para aumentar essa cobertura e, consequentemente, aumentar a confiança no código.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

4.4 TÉCNICAS DE TESTE BASEADAS NA EXPERIÊNCIA

FL-4.4.1 (K2) Explicar a suposição de erros.
FL-4.4.2 (K2) Explicar o teste exploratório.
FL-4.4.3 (K2) Explicar os testes baseados em listas de verificação.

4.4 TÉCNICAS DE TESTE BASEADAS NA EXPERIÊNCIA

- Casos de teste são derivados da **habilidade e intuição** do testador e de sua **experiência** com aplicativos e tecnologias semelhantes
- Podem ser úteis na identificação de testes que **não foram facilmente identificados por outras técnicas** mais sistemáticas
- A **cobertura pode ser difícil de avaliar** e pode não ser mensurável com essas técnicas

4.4.1 SUPOSIÇÃO DE ERRO

A suposição de erro é uma técnica usada para **prever a ocorrência de erros, defeitos e falhas**, com base no **conhecimento do testador**, incluindo:

- Como o aplicativo funcionou no passado;
- Que tipos de erro tendem a ser cometidos;
- Falhas ocorridas em outros aplicativos.

Criar uma lista de possíveis erros, defeitos e falhas e modelar os testes que exporão essas falhas e os defeitos que as causaram.

Podem ser construídos **com base na experiência, nos dados de defeitos e falhas ou no conhecimento comum sobre o motivo da falha** do software.

4.4.2 TESTE EXPLORATÓRIO

Nesta técnica, os testes informais (não pré-definidos) são modelados, executados, registrados e avaliados dinamicamente durante a execução do teste.

- Os resultados do teste são usados para **aprender mais sobre o componente ou sistema** e para **criar testes para as áreas que podem precisar** de mais testes.
- Às vezes, o teste exploratório é realizado usando **testes baseados em sessão** para estruturar as atividades de teste.
- No teste baseado em sessão, o teste exploratório é conduzido **dentro de uma janela de tempo definida**, e o testador usa um termo de teste contendo **objetivos de teste para orientar o teste**.
- O teste exploratório é mais útil quando há **poucas ou inadequadas especificações ou pressão de tempo** significativa nos testes.
- O teste exploratório também é útil para **complementar outras técnicas** de teste mais formais.

4.4.3 TESTE BASEADO EM LISTAS DE VERIFICAÇÃO (CHECKLIST)

Os testadores modelam, implementam e executam testes para cobrir as condições de teste encontradas em uma lista.

- Testadores podem **criar, expandir ou usar uma lista** (checklist) existente sem modificá-la.
- Checklists **podem ser construídos com base na experiência**, conhecimento sobre o que é importante para o usuário ou uma compreensão de por que e como o software falha.
- Na ausência de casos de teste detalhados, os testes baseados em checklist **podem fornecer diretrizes e consistência**
- Como essas listas são de **alto nível**, é provável que ocorra alguma **variabilidade nos testes reais**, resultando em uma **cobertura potencialmente maior**, mas com **menos repetibilidade**.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

4.5 ABORDAGENS DE TESTE BASEADAS NA COLABORAÇÃO

FL-4.5.1 (K2) Explicar como escrever histórias de usuários em colaboração com desenvolvedores e representantes de negócio.
FL-4.5.2 (K2) Classificar as diferentes opções para escrever critérios de aceite.
FL-4.5.3 (K3) Usar o desenvolvimento orientado por testes de aceite (ATDD) para derivar casos de teste.

4.5.1 ESCRITA COLABORATIVA DE HISTÓRIAS DE USUÁRIOS

As histórias de usuários têm três aspectos principais, chamados de **conjunto "3C"**:

- **Cartão**: o meio onde se descreve uma história de usuário (ex: um cartão de registro, uma entrada em um quadro eletrônico);
- **Conversação**: a explicação de como o software será usado (pode ser documentado ou verbal)
- **Confirmação**: os critérios de aceite

4.5.1 ESCRITA COLABORATIVA DE HISTÓRIAS DE USUÁRIOS

"**Como** [ator ou persona], **quero** [meta a ser cumprida], **para que eu possa** [valor de negócio resultante para a função]”,

4.5.1 ESCRITA COLABORATIVA DE HISTÓRIAS DE USUÁRIOS

A autoria colaborativa da história do usuário pode usar técnicas como **brainstorming** e **mapeamento mental**.

A colaboração permite que a equipe obtenha uma **visão compartilhada do que deve ser entregue**, levando em conta três perspectivas: **negócios, desenvolvimento e testes**.

4.5.1 ESCRITA COLABORATIVA DE HISTÓRIAS DE USUÁRIOS

Boas histórias de usuários devem ser:
Independentes, Negociáveis, Valiosas, Estimáveis, pequenas (Small) e Testáveis (**INVEST**).

Se um stakeholder não souber como testar uma história de usuário, isso pode indicar que a história de usuário não está suficientemente clara, ou que não reflete algo valioso para ele, ou que o stakeholder precisa apenas de ajuda para testar.

4.5.2 CRITÉRIOS DE ACEITE

Os critérios de aceite de uma história de usuário são as **condições que a implementação dela deve atender para ser aceita** pelos stakeholders

Critérios de aceite **podem ser vistos como as condições de teste** que devem ser executadas pelos testes

Os critérios de aceite **geralmente são resultado da Conversação**

4.5.2 CRITÉRIOS DE ACEITE

Eu, **como** Aluno do ensino médio, **gostaria de** enviar áudio explicando minha dúvida para o professor, **porque** eu gasto menos tempo do que escrevendo.

Critérios de Aceite:

- Arquivos de áudio nos formatos WAV, MP3 e MPG4;
- Arquivos de áudio de até 100mb;
- Permitir que o aluno ouça o áudio enviado;
- Não permitir que o áudio enviado seja excluído;

4.5.2 CRITÉRIOS DE ACEITE

Os critérios de aceite são usados para:

- Definir o escopo da história do usuário;
- Chegar a um consenso entre os stakeholders;
- Descrever os cenários positivos e negativos;
- Servir como base para o teste de aceite da história do usuário;
- Permitir planejamento e estimativa precisos.

4.5.2 CRITÉRIOS DE ACEITE

Há várias maneiras de escrever critérios de aceite para uma história de usuário. Os dois formatos mais comuns são:

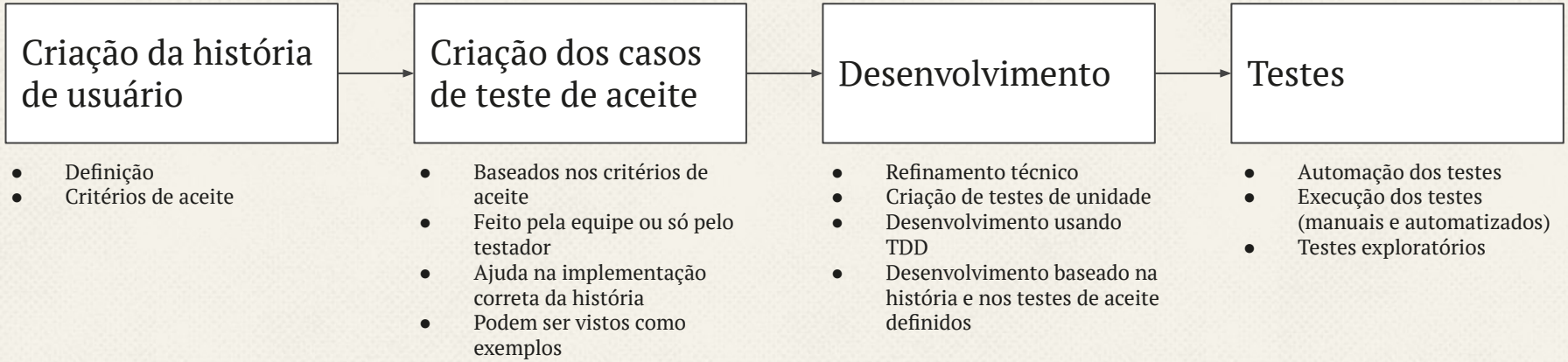
- **Orientado a cenários** (ex: formato Given/When/Then usado no BDD);
- **Orientado por regras** (ex: lista de pontos de verificação ou forma tabulada de mapeamento de entrada e saída);

A maioria dos critérios de aceite podem ser documentadas em um desses dois formatos.
No entanto, **a equipe pode usar outro formato personalizado**, desde que os critérios de aceite sejam bem definidos e não ambíguos.

4.5.3 DESENVOLVIMENTO ORIENTADO POR TESTE DE ACEITE (ATDD)

- O ATDD é uma **abordagem que prioriza o teste**.
- Os **casos de teste são criados antes da implementação** da história do usuário.
- Eles são **criados por membros da equipe com diferentes perspectivas**, por exemplo, clientes, desenvolvedores e Testadores, podendo ser executados de forma **manual ou automatizada**.

4.5.3 DESENVOLVIMENTO ORIENTADO POR TESTE DE ACEITE (ATDD)



Sobre casos de teste no ATDD

- Testes **positivos primeiro, depois testes negativos**
- Deve cobrir **características não funcionais**
- Devem ser **compreensíveis para stakeholders** (linguagem natural)
- Devem **abranger todas as características da história** (não ir além dela)
- Se escritos em formato compatível com automação, os testes de aceite tornam-se **requisitos executáveis**.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

CAPÍTULO 5

Gerenciamento das Atividades de Teste

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

5.1 PLANEJAMENTO DE TESTE

FL-5.1.1 (K2) Exemplificar a finalidade e o conteúdo de um plano de teste.
FL-5.1.2 (K1) Reconhecer como um Testador agrega valor ao planejamento de iteração e lançamento.
FL-5.1.3 (K2) Comparar e contrastar critérios de entrada e critérios de saída.
FL-5.1.4 (K3) Usar técnicas de estimativa para calcular o esforço necessário de teste.
FL-5.1.5 (K3) Aplicar a priorização de casos de teste.
FL-5.1.6 (K1) Relembrar os conceitos da pirâmide de teste.
FL-5.1.7 (K2) Resumir os quadrantes de teste e suas relações com os níveis e tipos de teste.

5.1.1 OBJETIVO E CONTEÚDO DE UM PLANO DE TESTE

Um plano de teste descreve os objetivos, os recursos e os processos de um projeto de teste.

- **Documenta os meios e o cronograma** para atingir os objetivos do teste;
- Ajuda a **garantir que as atividades de teste realizadas atenderão aos critérios** estabelecidos;
- Serve como um **meio de comunicação** com os membros da equipe e outros stakeholders;
- Demonstra que os **testes seguirão a política e a estratégia** de testes existentes (ou explica por que os testes se desviarão delas);

5.1.1 OBJETIVO E CONTEÚDO DE UM PLANO DE TESTE

O planejamento de testes **orienta o pensamento dos Testadores** e os força a enfrentar os desafios futuros relacionados a riscos, cronogramas, pessoas, ferramentas, custos, esforços, etc.

5.1.1 OBJETIVO E CONTEÚDO DE UM PLANO DE TESTE

O conteúdo típico de um plano de teste inclui:

- Contexto do teste (escopo, objetivos do teste, restrições, base do teste);
- Premissas e restrições do projeto de teste;
- Stakeholders (funções, responsabilidades, relevância para os testes, necessidades de contratação e treinamento);
- Comunicação (formas e frequência de comunicação, modelos de documentação);
- Registro de riscos (riscos do produto, riscos do projeto);
- Abordagem de teste (níveis, tipos, técnicas, produtos de teste, critérios de entrada e de saída, independência do teste, métricas a serem coletadas, requisitos de dados de teste, requisitos de ambiente de teste, desvios da política de teste organizacional e da estratégia de teste);
- Orçamento e cronograma.

5.1.2 CONTRIBUIÇÃO DO TESTADOR PARA O PLANEJAMENTO DE ITERAÇÃO E LIBERAÇÃO

PLANEJAMENTO DE LIBERAÇÃO (PREVÊ O LANÇAMENTO DE UMA NOVA VERSÃO)

- Participam da escrita das histórias de usuário
- Participam da escrita de critérios de aceite testáveis
- Participam das análises de risco do projeto e da qualidade
- Estimam o esforço de teste associado às histórias de usuário
- Determinam a abordagem de teste
- Planejam o teste para a versão

5.1.2 CONTRIBUIÇÃO DO TESTADOR PARA O PLANEJAMENTO DE ITERAÇÃO E LIBERAÇÃO

PLANEJAMENTO DE ITERAÇÃO (PREVÊ O FIM DE UMA ITERAÇÃO)

- Participam da análise de risco detalhada das histórias de usuários
- Determinam a testabilidade das histórias
- Dividem as atividades de teste em tarefas
- Estimam o esforço de teste para todas as tarefas de teste
- Identificam e refinam os aspectos funcionais e não funcionais do objeto de teste

5.1.3 CRITÉRIOS DE ENTRADA E CRITÉRIOS DE SAÍDA

Critérios de Entrada (DoR - Definition of Ready ou Definição de Pronto)

Tudo o que deve estar OK para que os testes possam iniciar.

Critérios típicos de entrada:

- Disponibilidade de recursos (pessoas, ferramentas, ambientes, dados de teste, orçamento, tempo)
- Disponibilidade de material de teste (base de teste, requisitos testáveis, histórias de usuários, casos de teste)
- Nível de qualidade inicial de um objeto de teste (ex: todos os testes de fumaça foram aprovados)

Critérios de Saída (DoD - Definition of Done ou Definição de Feito)

O que devem ser atingido para declarar que um nível de teste ou um conjunto de testes foram concluídos.

Critérios típicos de saída:

- Medidas de precisão (ex: nível de cobertura alcançado, número de defeitos não resolvidos, densidade de defeitos, número de casos de teste com falha)
- Critérios de conclusão (p. ex., testes planejados foram executados, testes estáticos foram realizados, todos os defeitos encontrados foram relatados, todos os testes de regressão foram automatizados)
- O esgotamento do tempo ou do orçamento

Mesmo sem que todos critérios de saída sejam atendidos, pode ser aceitável encerrar os testes, se os stakeholders tiverem analisado e aceitado o risco de entrar em operação sem mais testes

5.1.4 TÉCNICAS DE ESTIMATIVA

Técnica Baseada em Índices

O esforço do teste é estimado com base nas **métricas de projetos anteriores.**

Exemplo

- Se no projeto anterior a proporção de esforço de desenvolvimento para teste foi de 3:2 e no projeto atual espera-se que o esforço de desenvolvimento seja de 600 homens/hora, o esforço de teste pode ser estimado em 400 homens/hora.
- Basicamente regra de três para calcular:

600—--------3
X—---------2

600x2 = 1200
1200/3 = 400

Extrapolação

Baseada em métricas, coleta-se dados de medições o mais cedo possível, e com eles pode-se estimar o trabalho restante por meio da extrapolação desses dados (adicionando um tempo/esforço a mais). Muito adequado em modelos iterativos.

Exemplo

- A equipe pode extrapolar o esforço de teste na próxima iteração como o esforço médio das três últimas iterações.

5.1.4 TÉCNICAS DE ESTIMATIVA

Wideband Delphi

Especialistas estimam com base na experiência.
Cada especialista estima isoladamente.
Resultados são comparados, e discutem as divergências até chegar em um consenso.

Exemplo

- Planning Poker

Estimativa de três pontos

Baseada em especialistas.
3 estimativas são feitas (otimista (o), mais provável (m), e pessimista (p)).
A estimativa final (E) é a média aritmética ponderada dessas estimativas.
Também é possível calcular o erro de medição (SD).

$$E = \frac{o + 4m + p}{6} \qquad SD = \frac{(p - o)}{6}$$

Por exemplo, se as estimativas (em homens/hora) forem: o=6, m=9 e p=18, então a estimativa final é de 10±2 homens/hora (ou seja, entre 8 e 12 homens/hora), porque:

$$E = \frac{6+36+18}{6} = 10 \quad \text{e} \quad SD = \frac{(18-6)}{6} = 2$$

5.1.5 PRIORIZAÇÃO DE CASOS DE TESTE

ESTRATÉGIAS DE PRIORIZAÇÃO

- **Priorização baseada em risco**: a ordem de execução do teste é baseada nos resultados da análise de risco. Os casos de teste que abrangem os riscos mais importantes são executados primeiro.
- **Priorização baseada em cobertura**: a ordem de execução do teste é baseada na cobertura. Os casos de teste que atingem a maior cobertura são executados primeiro.
- **Priorização baseada em requisitos**: a ordem de execução do teste é baseada nas prioridades dos requisitos. Os casos de teste relacionados aos requisitos mais importantes são executados primeiro.

5.1.5 PRIORIZAÇÃO DE CASOS DE TESTE

Casos de teste devem ser **ordenados para execução com base em seus níveis de prioridade**.

Exceto se os casos de teste tiverem dependências. Se um caso de teste com prioridade mais alta for dependente de um caso de teste com prioridade mais baixa, o caso de teste de prioridade mais baixa deverá ser executado primeiro.

A ordem de execução do teste **também deve levar em conta a disponibilidade de recursos**. Por exemplo, as ferramentas de teste necessárias, os ambientes de teste ou as pessoas que podem estar disponíveis apenas em um período específico.

5.1.6 PIRÂMIDE DE TESTE

É um modelo que mostra que diferentes testes podem ter diferentes granularidades.

O modelo da pirâmide de testes **apoia a equipe na automação de testes e na alocação de esforço de teste**, mostrando que diferentes objetivos são apoiados por diferentes níveis de automação de testes.

As camadas da pirâmide representam grupos de testes. **Quanto mais alta a camada, menor a granularidade do teste, o isolamento do teste e o tempo de execução do teste.**

By Leonardo Carvalho

5.1.6 PIRÂMIDE DE TESTE

5.1.7 QUADRANTES DE TESTE

Agrupam os níveis de teste com os tipos de teste, as atividades, as técnicas de teste e os produtos de trabalho apropriados no desenvolvimento ágil.

O modelo ajuda o gerenciamento de testes a visualizá-los para **garantir que todos os tipos e níveis de teste apropriados sejam incluídos no SDLC** e a entender que alguns tipos de teste são mais relevantes para determinados níveis de teste do que outros.

Esse modelo também **fornece uma maneira de diferenciar e descrever os tipos de testes para todos os stakeholders**, incluindo desenvolvedores, Testadores e representantes do negócio

By Leonardo Carvalho

5.1.7 QUADRANTES DE TESTE

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

5.2 GERENCIAMENTO DE RISCO

FL-5.2.1 (K1) Identificar o nível de risco usando a probabilidade de risco e o impacto do risco.
FL-5.2.2 (K2) Distinguir entre riscos de projeto e riscos de produto.
FL-5.2.3 (K2) Explicar como a análise de risco do produto pode influenciar o rigor e o escopo dos testes.
FL-5.2.4 (K2) Explicar quais medidas podem ser tomadas em resposta aos riscos analisados do produto.

5.2 GERENCIAMENTO DE RISCO

As principais atividades de gerenciamento de riscos são:

- **Análise de risco** (que consiste na identificação e avaliação de risco).
- **Controle de riscos** (que consiste na mitigação e monitoramento de riscos).

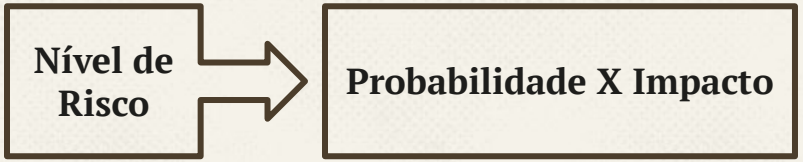
5.2.1 DEFINIÇÃO DE RISCO E ATRIBUTOS DO RISCO

- **Risco é um possível evento, perigo, ameaça ou situação** cuja ocorrência causa um efeito adverso.

- O **nível de risco** é determinado pela **probabilidade**, ou seja a chance de que aquilo venha ocorrer e pelo **impacto**, ou seja o dano causado por esse evento.

"Risco é um evento incerto, ou seja, não temos certeza se ele vai ou não ocorrer."

DECORAR!



5.2.2 RISCOS DO PROJETO E RISCOS DO PRODUTO

Risco de Produto

- Está diretamente relacionado com aquilo que você está testando
- Ex. Funcionalidade, Componente ou Produto
- Estão associados às características específicas de qualidade de um produto. **Também sendo chamados de riscos de qualidade.**

Operacional

Risco de Projeto

- Está diretamente relacionado ao gerenciamento do projeto
- Envolve situações que caso ocorram, podem ter um efeito negativo na capacidade de um projeto atingir seus objetivos

Gerencial (Atividades)

5.2.2 RISCOS DO PROJETO E RISCOS DO PRODUTO

By Leonardo Carvalho

Exemplos de Riscos de Projetos

Questões do Projeto	Questões Organizacionais	Questões Políticas	Questões Técnicas	Questões de Fornecedores
• Atrasos na entrega, na conclusão da tarefa ou na satisfação dos critérios de saída • Estimativas imprecisas, realocação de fundos para projetos de maior prioridade ou corte geral de custos na organização.	• Insuficiência de equipe, habilidades e treinamento • Questões pessoais podem causar problemas e conflitos. • Prioridades comerciais conflitantes podem causar indisponibilidade de usuários, equipe de negócios ou especialistas de assunto.	• Os testadores podem não comunicar adequadamente suas necessidades ou os resultados do teste. • Pode haver uma atitude imprópria em relação às expectativas de testes (Não apreciar o valor de encontrar defeitos).	• Má definição de Requisitos. • Ambiente de Teste não estar pronto no prazo. • Má gestão de defeitos pode resultar em defeitos acumulados e outras obrigações Técnicas.	• Um terceiro pode deixar de entregar um produto ou serviço necessário ou ir à falência. • Questões contratuais podem causar problemas ao projeto.

5.2.2 RISCOS DO PROJETO E RISCOS DO PRODUTO

Exemplos de Riscos de Produto ou Qualidade

- Insatisfação do usuário;
- Perda de receita, confiança e reputação;
- Danos a terceiros;
- Altos custos de manutenção, e sobrecarga do helpdesk;
- Penalidades criminais;
- Em casos extremos, danos físicos, lesões ou até mesmo a morte.

5.2.3 ANÁLISE DE RISCO DO PRODUTO

A análise de risco do produto consiste na **identificação e na avaliação de riscos**.

O objetivo é **proporcionar uma conscientização do risco do produto** para concentrar o esforço de teste de forma a minimizar o nível residual de risco do produto.

O ideal é que a análise comece no **início do SDLC**.

Técnicas: brainstorming, workshops, entrevistas ou diagramas de causa e efeito.

5.2.3 ANÁLISE DE RISCO DO PRODUTO

Pode usar uma abordagem quantitativa ou qualitativa, ou uma combinação delas.
Na quantitativa, o nível de risco é calculado como a **multiplicação da probabilidade e do impacto**.
Na qualitativa, o nível de risco pode ser determinado usando uma **matriz de risco**.

Seus resultados são usados para:

- Determinar o escopo dos testes;
- Determinar os níveis de teste e propor os tipos de teste a serem realizados;
- Determinar as técnicas e a cobertura a ser alcançada;
- Estimar o esforço de teste para cada tarefa;
- Priorizar os testes em uma tentativa de encontrar os defeitos críticos o mais cedo possível;
- Determinar se outras atividades, além dos testes, podem ser empregadas para reduzir o risco.

5.2.4 CONTROLE DE RISCO DO PRODUTO

O controle de riscos do produto consiste na **mitigação e no monitoramento dos riscos**.

Esta mitigação envolve a **implementação das ações propostas na avaliação de riscos para reduzir o nível de risco**.

O objetivo do monitoramento de riscos é:

- garantir que as **ações de mitigação sejam eficazes**,
- **melhorar a avaliação de riscos**
- identificar **riscos emergentes**.

5.2.4 CONTROLE DE RISCO DO PRODUTO

Uma vez que um risco tenha sido analisado, várias opções de resposta ao risco são possíveis, por exemplo, **mitigação do risco por meio de testes, aceite do risco, transferência do risco ou plano de contingência.**

As ações que podem ser tomadas para mitigar os riscos do produto por meio de testes:

- Selecionar os Testadores com o nível de experiência e habilidade adequado;
- Aplicar um nível adequado de independência de testes;
- Conduzir revisões e realizar análises estáticas;
- Aplicar as técnicas de teste e os níveis de cobertura adequados;
- Aplicar os tipos de teste apropriados que abordam as características afetadas;
- Realizar testes dinâmicos, incluindo testes de regressão.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

5.3 MONITORAMENTO, CONTROLE E CONCLUSÃO DO TESTE

FL-5.3.1 (K1) Métricas de recuperação usadas para testes.
FL-5.3.2 (K2) Resumir as finalidades, o conteúdo e o público dos relatórios de teste.
FL-5.3.3 (K2) Exemplificar como comunicar o status do teste.

5.3 MONITORAMENTO, CONTROLE E CONCLUSÃO DO TESTE

Informações coletadas no monitoramento são **usadas para avaliar o progresso do teste e medir se os critérios de saída do teste foram atendidos**.

O controle de testes usa as informações do monitoramento de testes para fornecer orientação e as ações corretivas necessárias.

Exemplos de diretrizes de controle:

- Repriorizar os testes quando um risco se torna um problema;
- Reavaliar se um item de teste atende aos critérios de entrada ou saída devido ao retrabalho;
- Ajustar o cronograma de testes para lidar com um atraso na entrega do ambiente de teste;
- Adicionar novos recursos quando e onde forem necessários.

5.3 MONITORAMENTO, CONTROLE E CONCLUSÃO DO TESTE

A conclusão do teste **coleta dados de atividades de teste concluídas** para consolidar a experiência, o material de teste e qualquer outra informação relevante.

As atividades de conclusão do teste ocorrem em marcos do projeto, como quando um nível de teste é concluído, uma iteração Ágil é finalizada, um projeto de teste é concluído (ou cancelado), um sistema de software é lançado ou uma versão de manutenção é concluída.

5.3.1 MÉTRICAS USADAS EM TESTES

As métricas podem ser coletadas **durante o fim das atividades do teste** com objetivo de avaliar os seguintes itens:

- Relação entre o **planejado e o orçado** em um cronograma.
- **Qualidade** atual do objeto de teste.
- **Rendimento** das atividades de teste em relação aos objetivos.

Exemplos de **Métricas** de Teste:

- **Métricas de progresso do projeto** (ex: conclusão de tarefas, uso de recursos, esforço de teste);
- **Métricas de progresso do teste** (ex: progresso da implementação do caso de teste, progresso da preparação do ambiente de teste, número de casos de teste executados/não executados, aprovados/fracassados, tempo de execução do teste);
- Métricas de **qualidade do produto** (ex: disponibilidade, tempo de resposta, tempo médio até a falha);
- **Métricas de defeitos** (ex: número e prioridades de defeitos encontrados/corrigidos, densidade de defeitos, porcentagem de detecção de defeitos);
- **Métricas de risco** (ex: nível de risco residual);
- **Métricas de cobertura** (ex: cobertura de requisitos, cobertura de código);
- **Métricas de custo** (ex: custo de teste, custo organizacional de qualidade).

5.3.2 RELATÓRIOS DE TESTE: OBJETIVO, CONTEÚDO E PÚBLICO-ALVO

OBJETIVO
Os relatórios de teste **resumem e comunicam as informações do teste** durante e após o teste.

Os relatórios de progresso do teste dão **suporte ao controle contínuo do teste** e devem fornecer informações suficientes para fazer modificações no cronograma, nos recursos ou no plano de teste, quando essas alterações forem necessárias devido a desvios do plano ou mudanças nas circunstâncias.

Os relatórios de conclusão do teste **resumem um estágio específico do teste** (p. ex., nível de teste, ciclo de teste, iteração) e podem fornecer informações para testes subsequentes.

5.3.2 RELATÓRIOS DE TESTE: OBJETIVO, CONTEÚDO E PÚBLICO-ALVO

CONTEÚDO

Relatório de Progresso geralmente são gerados regularmente (p. ex., diariamente, semanalmente etc.) e incluem:

- Período de teste;
- Progresso do teste (p. ex., adiantado ou atrasado);
- Impedimentos para testes e suas soluções;
- Métricas de teste;
- Riscos novos e alterados durante o período de teste;
- Testes planejados para o próximo período.

5.3.2 RELATÓRIOS DE TESTE: OBJETIVO, CONTEÚDO E PÚBLICO-ALVO

CONTEÚDO

Um **Relatório de Conclusão** de teste é preparado durante a conclusão do teste, quando um projeto, nível de teste ou tipo de teste está concluído e quando, idealmente, **seus critérios de saída foram atendidos**. Incluem:

- Resumo do teste;
- Testes e avaliação da qualidade do produto com base no plano de teste original (ou seja, objetivos de teste e critérios de saída);
- Desvios do plano de teste (p. ex., diferenças em relação ao cronograma, à duração e ao esforço planejados);
- Impedimentos e soluções alternativas para o teste;
- Métricas de teste baseadas em relatórios de progresso de teste;
- Riscos não mitigados, defeitos não corrigidos;
- Lições aprendidas que são relevantes para o teste;

5.3.2 RELATÓRIOS DE TESTE: OBJETIVO, CONTEÚDO E PÚBLICO-ALVO

PÚBLICO-ALVO

- **Públicos diferentes exigem informações diferentes** nos relatórios e influenciam o grau de formalidade e a frequência dos relatórios.
- Os relatórios sobre o progresso dos testes para outras pessoas da mesma equipe **costumam ser frequentes e informais**, enquanto os relatórios sobre os testes de um projeto concluído **seguem um modelo definido** e ocorrem apenas uma vez

5.3.3 COMUNICAÇÃO DO STATUS DOS TESTES

A forma de comunicação **varia de acordo com a necessidade**. Algumas opções:

- **Comunicação verbal** com membros da equipe e outros stakeholders;
- **Painéis** (p. ex., painéis de CI/CD, painéis de tarefas e gráficos de burn-down);
- Canais de **comunicação eletrônica** (p. ex., e-mail, bate-papo);
- **Documentação on-line**;
- **Relatórios de testes formais**.

Uma ou mais dessas opções podem ser usadas.

Uma comunicação mais formal pode ser mais apropriada para equipes distribuídas, em que a comunicação direta, face a face, nem sempre é possível devido à distância geográfica ou às diferenças de horário.

Normalmente, diferentes stakeholders estão interessados em diferentes tipos de informações, portanto, a comunicação deve ser adaptada de acordo.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

5.4 GERENCIAMENTO DE CONFIGURAÇÃO (CM)

FL-5.4.1 (K2) Resumir como o gerenciamento de configuração apoia os testes

5.4 GERENCIAMENTO DE CONFIGURAÇÃO (CM)

Tem como objetivo **estabelecer e manter a integridade do componente ou do sistema**, do testware e seus relacionamentos entre si durante o ciclo de vida do projeto e do produto.

Identificar, controlar e rastrear produtos de trabalho, como planos de teste, estratégias de teste, condições de teste, casos de teste, scripts de teste, resultados de teste, registros de teste e relatórios de teste como itens de configuração.

O gerenciamento de configuração **mantém um registro dos itens de configuração alterados** quando uma nova baseline é criada.
É possível reverter para uma baseline anterior para reproduzir resultados de testes anteriores.

5.4 GERENCIAMENTO DE CONFIGURAÇÃO (CM)

Para dar suporte adequado aos testes, o CM garante o seguinte:

- Todos os itens de configuração, incluindo itens de teste, **são identificados de forma exclusiva**, controlados por versão, rastreados quanto a alterações para que a **rastreabilidade** possa ser mantida durante todo o processo de teste
- Todos os itens de documentação e software identificados são referenciados **sem ambiguidade na documentação de teste**

A integração contínua, a entrega contínua, a implantação contínua e os testes associados normalmente são implementados como parte de um pipeline de DevOps automatizado, **no qual a CM automatizada normalmente está incluída**.

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

5.5 GERENCIAMENTO DE DEFEITOS

FL-5.5.1 (K3) Preparar um relatório de defeitos.

5.5 GERENCIAMENTO DE DEFEITOS

- As anomalias (não somente defeitos em si) **podem ser relatadas durante qualquer fase** do SDLC
- O processo de gerenciamento de defeitos inclui um **fluxo de trabalho para lidar com anomalias**, desde sua descoberta até seu fechamento, e regras para sua classificação.
- Alguns **dados podem ser incluídos automaticamente ao usar ferramentas** de gerenciamento de defeitos

Objetivos dos Relatórios de Defeitos (Registro de Defeitos)

- Fornecer **informações suficientes para resolver o problema**.
- Fornecer meios **de rastrear a qualidade do produto de trabalho e o impacto no teste**.
- Fornecer **ideias para aprimoramento** dos processos de teste.

5.5 GERENCIAMENTO DE DEFEITOS

Conteúdo de um Relatório de Defeitos (Registro de Defeitos)

Identificador – Ex. Código/ID

Título/Resumo – Ex. Erro ao Cadastrar Produto de Venda: Sistema apresenta erro de verifique sua conexão ao cadastrar um produto de venda que esteja marcado como opcional

Data/Autor – Ex. 20/01/2019 Criado por Michele Prado

Identificação do Objeto de Teste e Ambiente – Ex. Módulo Produto de Venda - Ambiente de Desenvolvimento

Contexto do defeito – Ex. Testes Regressivos (caso de teste que está sendo executado, atividade de teste que está sendo realizada, fase do SDLC e outras informações relevantes, como a técnica de teste ou os dados de teste)

Descrição do Defeito – Ex. Capturas de Tela / Evidências em Vídeo / Logs

Resultados Esperados e Reais – Ex. Produto cadastrado / Sistema apresentou mensagem de erro

Severidade – Severidade é o grau de impacto nos interesses dos stakeholders ou requisitos.

Prioridade – Prioridade de correção

Status – Ex. Aberto

Referências – Ex. Caso de teste relacionado

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

CAPÍTULO 6

Ferramentas de Teste

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

6.1 SUPORTE DE FERRAMENTAS PARA TESTES

FL-6.1.1 (K2) Explicar como os diferentes tipos de ferramentas de teste dão suporte aos testes.

6.1 SUPORTE DE FERRAMENTAS PARA TESTES

Exemplos de tipos de ferramentas de teste:

- **Ferramentas de gerenciamento**: aumentam a eficiência do processo de teste, facilitando o gerenciamento do SDLC, dos requisitos, dos testes, dos defeitos e da configuração;
- **Ferramentas de teste estático**: dão suporte ao Testador na realização de revisões e análises estáticas;
- **Ferramentas de projeto e implementação de testes**: facilitam a geração de casos de teste, dados de teste e procedimentos de teste;
- **Ferramentas de execução e cobertura de testes**: facilitam a execução de testes automatizados e a medição da cobertura;
- **Ferramentas de teste não funcional**: permitem que o Testador realize testes não funcionais que são difíceis ou impossíveis de serem realizados manualmente;
- **Ferramentas de DevOps**: suporte ao pipeline de entrega de DevOps, rastreamento de fluxo de trabalho, processo(s) de construção automatizado(s), CI/CD;
- **Ferramentas de colaboração**: facilitam a comunicação;
- **Ferramentas que oferecem suporte à escalabilidade e à padronização da implantação** (ex: máquinas virtuais, ferramentas de conteinerização);
- **Qualquer outra ferramenta que auxilie no teste** (ex: uma planilha pode ser uma ferramenta).

CTFL
CURSO PREPARATÓRIO

By Leonardo Carvalho

6.2 BENEFÍCIOS E RISCOS DA AUTOMAÇÃO DE TESTES

FL-6.2.1 (K1) Relembrar os benefícios e os riscos da automação de testes

6.2 BENEFÍCIOS E RISCOS DA AUTOMAÇÃO DE TESTES

Ao introduzir uma nova ferramenta em uma organização, será exigido um esforço para obter benefícios reais e duradouros. Existem benefícios e oportunidades potenciais com o uso de ferramentas de automação de testes, mas também há riscos.

Benefícios

- **Redução no trabalho manual repetitivo**, economizando tempo.
- **Maior consistência e repetibilidade** (testes são executados por uma ferramenta na mesma ordem com a mesma frequência)
- **Avaliação mais objetiva** (Medidas Estáticas, Cobertura)
- **Acesso mais fácil a informações sobre testes** (Estatísticas e gráficos sobre o progresso do teste, etc)
- **Redução dos tempos de execução** de testes para proporcionar detecção antecipada de defeitos, feedback mais rápido e menor tempo de lançamento no mercado
- **Mais tempo para os Testadores criarem testes novos**

6.2 BENEFÍCIOS E RISCOS DA AUTOMAÇÃO DE TESTES

Riscos

- **Expectativas irreais sobre os benefícios de uma ferramenta** (incluindo funcionalidade e facilidade de uso);
- **Estimativas imprecisas** de tempo, custos e esforço necessários para introduzir uma ferramenta, manter scripts de teste e alterar o processo de teste manual existente;
- **Usar uma ferramenta de teste quando o teste manual é mais apropriado;**
- **Confiar demais em uma ferramenta**, por exemplo, ignorando a necessidade do pensamento crítico humano;
- **A dependência do fornecedor da ferramenta**, que pode fechar as portas, aposentar a ferramenta, vender a ferramenta para outro fornecedor ou fornecer um suporte deficiente;
- **Usar um software de código aberto que pode ser abandonado**, o que significa que não há mais atualizações disponíveis, ou que seus componentes internos podem exigir atualizações bastante frequentes como um desenvolvimento adicional;
- **A ferramenta de automação não ser compatível** com a plataforma de desenvolvimento;
- **Escolha de uma ferramenta inadequada** que não cumpra os requisitos normativos e/ou as normas de segurança.