CTFL

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

Ajuda

Em algumas ramificações de tópicos existem hiperlink ou comentários para te ajudar a compreender melhor o tópico

<== Existem ramos com ícone de bandeira vermelha, significa que são assuntos, quentes, que provavelmente vai está na prova

Este Mapa mental não exclui o estudo pelo syllabus, é muito importante estudar toda a apostila do syllabus

Syllabus 3.1

Este Mapa mental não exclui o estudo por cursos de CFTL, é muito importante estudar por cursos voltados ao CTFL

Cursos

- Curso CTFL Iterasys
- Curso CTFL Leonardo Carvalho
- Curso CTFL Udemy melhor avaliado
- Curso playlist Pessoni

Resumo CTFL

Exemplos de Exame CTFL

- Estude os simulados, na sua prova pode conter algumas questões dos simulados
- Exemplo de Exame CTFL 01
- Exemplo de Exame CTFL 02
- Exemplo de Exame CTFL 03

Simulados

- Simulados 01
- Simulados 02
- Ajuda Técnicas de teste
  - Caixa preta
  - Caixa branca

Informações

- Informações sobre a prova

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- **Pré-requisitos**: nenhum
- **Número de questões**: 40
- **Tipo de questões**: múltipla escolha
- **Tempo de Exame***: 60 minutos
- **Pontuação**: 1 ponto por questão
- **Aprovação**: mínimo de 65% de acertos ou 26 pontos
- **Modalidades de Exame**: Acadêmico, Empresarial, Nacional, Especial, Treinamento e Online.
- **Valor do Exame**: ~~R$ 800,00~~ *(consulte os valores promocionais em cada modalidade de exame)*

- Informações Importantes

Informações Importantes

1. **Não é obrigatório** que o candidato passe por um **treinamento** para participar do exame de certificação. Caso o candidato necessite, sugerimos que o faça com um dos **Provedores de Treinamento Certificados**, que possuem treinamentos, específicos para esta certificação, que foram avaliados e aprovados pelo BSTQB.
2. Todas as questões dos exames na língua portuguesa foram criadas e revisadas pela equipe do **GT Exame** do BSTQB.
3. Seguindo padrões internacionais do ISTQB, **não divulgaremos o gabarito**, nem **realizaremos qualquer revisão** na correção dos exames. O processo é eletrônico com múltiplas conferências manuais garantindo que não haja erros de correção.
4. Para auxiliar o candidato sobre sua evolução no exame, o resultado encaminhado conterá os percentuais de acerto em cada capítulo.
5. As **questões do exame são distribuídas** proporcionalmente ao tempo de estudo de cada capítulo, proposto pelo ISTQB no syllabus. Pequenas variações podem ocorrer a cada prova já que a divisão não é exata.

CAPÍTULO	1º	2º	3º	4º	5º	6º
QUESTÕES	7	5	3	12	9	4

Glossário

- Glossário padrão de termos utilizados
- Base de teste
  - Todos os documentos dos quais os requisitos de um componente ou sistema podem ser deduzidos
- Carta de teste
  - Declaração dos objetivos do teste e de possíveis ideias sobre como realizar os testes. As cartas de teste são usadas em testes exploratórios.
- caso de uso
  - Sequência de transações em um diálogo entre um ator e um componente ou sistema, com um resultado tangível, onde um ator pode ser um usuário ou qualquer coisa que possa trocar informações com o sistema
- cenário de teste
  - Ver especificação de procedimento de teste
- ciclo de teste
  - Execução do processo de teste contra um único release identificável do objeto de teste
- cobertura

        - Grau, expresso como uma porcentagem, que indica o quanto um item de cobertura foi exercitado por uma suíte de testes
    - Folha de sessão
        - São utilizadas para registrar etapas do teste e registrar descobertas feitas
    - Objetivo do teste
        - Software, produto que vai ser testado
    - Procedimentos de testes
        - Passo a passo ou sites do caso de teste
    - Testware
        - Tudo que você é capaz de produzir
    - Time-Box
        - Janela de tempo pré definida
    - Caso de teste
        - Título e Descrição
    - Roteiro de teste
        - Passo a Passo a passo do caso de teste

Normas:

- ISO/IEC/IEEE 29119-1
    - Conceito de teste
- ISO/IEC/IEEE 29119-2
    - Processo de teste
- ISSO/IEC/IEEE 29119-3
    - Produtos de trabalho
- ISO / IEC / IEEE 29119-4
    - Técnicas de teste
- ISO20246
    - Processo de Revisão
- ISO25010
    - Características de qualidade

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

Criador: Thiago Maciel

Elogios, reclamações ou colaboração

- Email: maciel.thiago@gmail.com
- Linkedin: Thiago Andrade

Capitulo 01 Fundamentos do Teste

O que é teste

- Processo de teste inclui execução de teste, planejamento de testes, analise, modelagem e implementação dos testes, relatórios de progresso e resultados de testes e avaliação da qualidade de um objeto de teste (Software ou Software Alvo ou SUT).
- Teste dinâmico: Execução de um componente ou sistema
    - Principais níveis de teste: Componente, Integração, Sistema e Aceite (CISA).
        - Componente, teste em um único componente do sistema. Ex: Teste de unidade Ramo de Tópico
        - Integração, teste de 2 ou mais componentes do sistema (Integração pequena) e Integração de sistemas (Integração grande). Ex: Teste de APIRamo de Tópico
        - Sistema, teste do software como um todo. Ex: Teste de sistema camada de usuário, mobile, desktop...Tópico
        - Aceite ou Homologação, teste na visão do usuário ou cliente. Ex: Teste Usuário ou cliente que homologa o software.
- Teste Estático: Teste não envolvem a execução do componente ou sistema (Checagem de documentação, comparar código, verificar padrão).
- Teste também inclui revisão de produtos de trabalho, requisitos, histórias de usuário e código fonte
- Base de teste: Tudo que usamos de informação pra testar. Todos os documentos dos quais os requisitos de um componente ou sistema podem ser deduzidos. A documentação na qual os casos de testes são baseados.
- Testware: Tudo que você é capaz de produzir. Ex: Caso de teste, relatório, Script de teste.
- Objetivos típicos do teste:
    - Avaliar os produtos: Requisitos, estórias, modelagem e código.
    - Validar se o objeto de teste está completo e funciona como o usuário espera.
    - Criar confiança no nível de qualidade do objeto de teste.
    - Evitar Defeitos
    - Encontrar falhas e defeitos

        - Fornecer informações suficientes as partes interessadas em relação ao nível de qualidade
        - Reduzir nível de risco de qualidade de software (Falhas não detectadas anteriormente que ocorrem em produção)
        - Cumprir com requisitos ou normas contratuais e verificar o cumprimento do objeto de teste
        - Os objetivos do teste podem variar, dependendo do contexto do componente ou sistema, nível de teste e do modelo de ciclo de vida de software:
            - Durante o teste do componente o objetivo encontrar falhas o quanto possível e aumentar a cobertura de código dos testes de componentes, normalmente é feito pelo desenvolvedor. Ex: Teste unidade.
            - Teste de aceite tem objetivo confirmar que sistema funciona como esperado e satisfaz os requisitos.
    - Teste e depuração de código
        - Teste
            - Mostra falhas causadas por defeitos (bugs) no software
                - Os testes de confirmação subsequentes verificam se as correções resolveram os defeitos.
                    - Diferença:
                        - Teste de confirmação (reteste): Teste pontual naquele item ou funcionalidade com defeito
                        - Teste de Regressão: Teste geral no sistema, normalmente em uma nova implementação ou melhoria.
            - Normalmente feita pelo testador
        - Depuração
            - A depuração do código é a atividade de desenvolvimento que localiza, analisa e corrige esses defeitos
                - Teste de componente associado
            - Normalmente feito pelos desenvolvedores, mas o testador pode participar
        - A execução dos testes pode mostrar falhas causadas por defeitos, normalmente feito pelo testador.
        - Dentro do desenvolvimento ágil não existe é divisão de papel, todos podem estar envolvidos na depuração e no teste de componentes.
    - (ISO/IEC/IEEE 29119-1) Conceito de teste
    - O teste de software é uma maneira de avaliar a qualidade do software e reduzir o risco de falha de software em operação

Porque o teste é necessário

- Quando defeitos são detectados e corrigidos existe uma contribuição para a qualidade do sistema
- Testes rigorosos podem ajudar a reduzir o risco de falhas.
- Contribuição do teste para o sucesso
    - Ao longo da história da computação é bastante comum que os softwares sejam entregues a operação e devido a presença de defeitos subsequentemente causar falhas ou não atender as necessidades.
    - No entanto com o uso de técnicas de teste pode-se reduzir esses problemas de entrega:
        - Ter testadores envolvidos nas revisões de requisitos.
            - Shift left: Teste iniciam mais cedo e duram todo o ciclo de vida do desenvolvimento do software
        - Ter testadores trabalhando em conjunto com os projetistas do sistema, pode aumentar o entendimento de cada parte sobre o projeto
        - Ter testadores trabalhando em estreita colaboração com os desenvolvedores.
        - Ter testadores para verificar e validar o software.
- Teste pode ser necessário para atender os requisitos contratuais ou legais ou aos padrões específicos. Cumprir contratos.
- Garantia da qualidade e teste
    - A gestão da qualidade inclui todas as atividades que direcionam e controlam uma organização em relação a qualidade.
    - A gestão da qualidade inclui a garantia de qualidade e o controle de qualidade. Garantir um produto mais estável, melhorar processo, mudança de ferramentas...
    - A garantia de qualidade é tipicamente focada na adesão a processos adequados, a fim de fornecer confiança de que os níveis apropriados de qualidade serão lançados
    - Quando os processos são realizados adequadamente, contribui para a prevenção de defeitos
    - Analise de causa raiz para detectar e remover as causas de defeitos
- Erros, defeitos e falhas
    - Erro (Engano): Pode somente achar um erro ou engano a própria pessoa que gerou. Ex: Analista de teste encontrar algo incorreto no seu caso de teste isso é um erro ou engano.
        - • Pressão humana • Falha humana • Participante do projeto inexperientes • Falta de comunicação • Complexidade do código • mal-entendidos-entendidos sobre interfaces • Tecnologias novas.
    - Defeito ou falha ou Bug: Outra Pessoa encontrou o Defeito ou falha ou Bug.

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

        - Defeito: Durante um teste estático (Revisão documento, código... sem executar por outra pessoa sem ser o autor).
        - Falha: Durante um teste dinâmico (Executado por outra pessoa sem ser o autor)
    - Um erro leva a um defeito que pode virar uma falha
    - Pode causar uma falha, mas não necessariamente um defeito. Ex: Falhas podem ser causadas por condições ambientais.
    - Podem ocorrer testes:
        - Falso negativos: São testes que não detectam defeitos que deveriam ser detectados. Ex: Pode ocorrer uma situação muito especificas, teste de calculadora 2 + 2 = 4 e 2 x 2 = 4.
        - Falso positivo: São relatados como defeitos, mas na verdade não são defeitos. Ex: Alguma integração desligada
- Defeitos, causas-raiz e efeitos
    - As causas-raiz dos defeitos são as primeiras ou condições que contribuíram para criação de defeitos
    - Ao identificar a causa-raiz pode reduzir a ocorrência de defeitos futuros similares.

Sete princípios de teste

- O teste mostra a presença de defeitos e não a sua ausência:
    - O teste reduz a probabilidade de defeitos não descobertos permanecerem no software, o teste não é uma prova de correção.
- Testes exaustivos são impossíveis
    - Testar tudo, não é viável, as técnicas de teste e as prioridades devem ser usadas para concentrar os esforços de teste.
- O teste inicial economiza tempo e dinheiro
    - Teste Estático e dinâmico deve iniciar o mais cedo possível. Shift Left
- Defeitos se agrupam
    - Um pequeno número dos módulos geralmente contém a maioria dos defeitos.
- Cuidado com o paradoxo do pesticida
    - Se os mesmos teste forem repetidos várias vezes, esses testes não encontrarão novos defeitos. Testes não são mais eficazes em encontrar defeitos, assim como pesticidas não são eficazes em matar insetos ao longo do tempo
        - Acabar com o paradoxo sempre criar novos teste e aumentar e melhorar a cobertura
- O teste depende do contexto

        - O teste é feito de forma diferente em diferente contexto.
            - Teste mais importante que outros. É mais importante testar um software de avião do que um software de loja
    - Ausência (Falácia) de erros é uma ilusão
        - Esperar que apenas encontrar e corrigir muitos defeitos assegure o sucesso de um sistema.

Processo de teste

- Não existe um processo universal de teste de software, mas há conjuntos comuns de atividades de teste
- Conjuntos de atividades são um processo de teste.
- Processo de teste de software especifico depende de muitos fatores.
- Esse conjunto de atividades serão discutidos na estratégia de teste.
- Processo de teste no contexto (7 no total)
    - Processo de teste de uma organização:
        - Modelo de ciclo de vida de desenvolvimento de software
        - Metodologia de projetos utilizados
        - Níveis de teste e tipos de testes
        - Riscos de produto e projeto
        - Domínio do negócio, conhecimento do negocio
        - Algumas restrições operacionais.
            - Orçamentos e recursos, tempo complexidade, requisitos contratuais
        - Políticas e praticas
        - Normas internas e externas.
    - Processo de teste organizacionais:
        - Atividade e tarefas de teste
        - Produtos de trabalho de teste
        - Rastreabilidade entre a base de teste e os produtos de trabalho de teste.
    - É muito útil se a base de teste tiver definidas critérios de cobertura mensuráveis. O quanto está sendo testado. Indicadores chave de desempenho (KPIs).
        - KPIs: Key Perfomace Indicator, indicador chave de performance, é uma forma de medir se uma ação ou um conjunto de iniciativas está efetivamente atendendo aos objetivos.

        - Os critérios de cobertura podem exigir pelo menos um caso de teste para cada elemento da sua base de teste.
        - (ISO/IEC/IEEE 29119-2) Processo de teste
    - Atividades e tarefas de teste
        - Cada grupo de atividades é composto por atividades constituintes, que podem variar de um projeto ou lançamento para outro.
        - Atividades podem ser sobrepostas, podem acontecer simultaneamente, pode não conter uma etapa
            - Planejamento do teste
                - Envolve no que testar e a abordagem do teste para atender os objetivos do teste.
            - Monitoramento e controle de teste
                - Monitoramento: Controle de teste envolve a comparação continua do progresso real com o plano de teste usando qualquer métrica de monitoramento definida pelo plano de teste. Acompanhar uma ação.
                - Controle: Tomada de ações necessárias para atender aos objetivos do plano de teste. Realizar uma ação.
                - Verificar os resultados em relação aos critérios de cobertura
                - Avaliar o nível de qualidade do sistema
                - Determinar se são necessários, mais testes
                - Progresso de teste em relação ao plano é comunicado as partes interessadas.
            - Analise do teste
                - O que testar em termos dos critérios de cobertura mensuráveis.
                - Analisar a base de teste apropriado ao nível de teste (CISA):
                    - As especificações dos requisitos
                    - modelagem e a implementação de informações.
                    - implementação do componente ou sistema
                    - Relatórios de análise de risco.
                - Avaliar a base de teste para identificar os tipos de defeitos:
                    - Ambiguidades
                    - omissões
                    - inconsistência

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

        - imprecisões
        - contradições
        - declarações supérfluas.
    - Identificar os recursos e os conjuntos de recursos a serem testados.
    - Definir e priorizar as condições de teste para cada recurso
    - Capturar a rastreabilidade bidirecional.
    - Aplicação de técnicas de teste caixa preta, caixa branca pode ser útil no processo de análise do teste.
    - Cartas de testes são típicos produtos de trabalho em alguns tipos de testes baseados na experiencia (Exploratório).
    - A identificação dos defeitos durante a análise do teste e um benefício potencial importante.
    - Verificam se os requisitos são consistentes, expressos adequadamente completo, validam e os requisitos capturam adequadamente as necessidades do cliente.
        - BDD: Desenvolvimento orientado ao comportamento, visão do PO (Product Owner).
        - ATDD: Desenvolvimento orientado a testes de aceite, visão do cliente ou usuário.
- Modelagem do teste
    - Durante a modelagem de teste, as condições de testes são elaboradas em casos de teste de alto nível.
    - Modelagem de teste é como testar.
    - Projetar e priorizar casos de teste e conjuntos de caos de teste
    - Identificar dados de teste necessários para comportar as condições de teste e os casos de teste
    - Projetar o ambiente de teste
    - Capturar a rastreabilidade bidirecional
    - Elaboração de teste durante a modelagem muitas vezes o uso de técnica de teste
    - A identificação dos defeitos durante a análise do teste é um benefício potencial importante.
- Implementação do teste
    - Testware é necessário

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

    - Implementação “Agora temos tudo pra executar os testes”
    - Desenvolver e priorizar os procedimentos de testes
    - Criar suítes de teste (Conjunto de teste) a partir dos procedimentos de teste (passo a passo)
    - Organizar os conjuntos de teste dentro de um cronograma
    - Construir ambiente de teste
    - Preparar os dados de teste
    - Verificar e atualizar a rastreabilidade bidirecional
    - Normalmente é junto com a modelagem
    - Nos testes exploratórios e em outro tipo de testes baseado em experiência, a modelagem, implementação e execução são feitos simultaneamente.
        - Teste exploratório podem ser baseados em cartas de teste
- Execução do teste
    - São executados de acordo com a programação da execução do teste
    - Gravar os identificadores e versões do item de teste, objeto de teste, da ferramenta de teste de software
    - Executar os testes manualmente ou usando ferramentas de execução
    - Comprar os resultados reais com os esperados
    - Analisar anomalias para estabelecer suas prováveis causas
    - Comunicar os defeitos com base nas falhas observadas
    - Registrar o resultado da execução
    - Repetir as atividades de teste como resultado de uma ação tomada por uma anomalia
    - Verificar e atualizar a rastreabilidade bidirecional
- Conclusão do teste
    - Coletam os dados da atividade de teste já concluídas para consolidar a experiencia, o testware e qualquer outra informação relevante.
    - Todos os defeitos encontrados foram fechados e garantir que esses defeitos não fechados sejam registrados para serem resolvidos
    - Relatório de resumo de teste (release notes)
    - Finalizar e arquivar o ambiente de teste

        - Analisar as lições aprendidas
        - Usar as informações coletadas pra melhorar futuros processo de teste
- Produtos de trabalho de teste
    - Produtos de teste são criados como parte do processo de teste
        - Como há variação significativa na maneira como as organizações implementam o processo de teste, há também uma variação significativa nos tipos de produtos de trabalho criado durante esse processo.
        - (ISO/IEC/IEEE 29119-3) Produtos de trabalho do teste, testware.
        - Criados através de uma série de ferramenta que gerenciam os testes, defeitos e execução.
    - Produtos de trabalho do planejamento do teste
        - Produtos de trabalho de planejamento do teste geralmente incluem um ou mais planos de teste.
        - Plano de teste inclui informações sobre a base de teste, com as quais os outros produtos de teste serão relacionados através das informações de rastreabilidade bem como os critérios de saída, que serão usados durante monitoramento
    - Produtos de trabalho de monitoramento e controle de teste
        - Geralmente incluem vários tipos de relatórios, incluindo relatórios de progresso de teste.
        - Todos os relatórios devem fornecer detalhes relevantes do público sobre o progresso e resultados
        - Os produtos ajudam a definir mais recursos
    - Produtos de trabalho da análise do teste
        - Condições bem definidas e priorizadas, preferencialmente onde cada uma das quais é bidireccionalmente rastreável
        - Testes exploratórios, criação das cartas de teste
        - Teste especificação, escrever caso de teste
    - Produtos de trabalho da modelagem do teste
        - Resulta em casos de teste e conjuntos de casos de teste para exercer as condições de teste definidas na análise de teste.
        - Caso de teste com mais detalhes, não é os passos
        - Resulta no projeto ou na identificação dos dados necessários de teste
    - Produtos de trabalho da implementação do teste

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- Procedimentos de teste e seu sequenciamento (passos ou steps)
- Suítes de teste (Conjuntos de teste)
- Cronograma de execução do teste
- Uma vez que a implementação é concluída, vamos ter os critérios do que será testado.
- Em alguns casos a criação de ambiente e verificação dos dados Massa de dados X Massa de teste: Massa de teste precisa conter os valores de entra e saída.
    - Os dados de teste servem para atribuir valores concretos as entradas e aos resultados esperados da execução
- Nos testes exploratórios, alguns produtos de trabalho da modelagem e implementação do teste podem ser criados durante a execução
    - Teste exploratório
        - Planejamento (Montar uma carta de teste)
        - Time box: Tem um limite de tempo para ser executado (Normalmente 30 a 120 min)
        - Testador experiente
    - Teste Ad-hoc (Adioc)
        - Não tem planejamento (Testa aí)
        - Não existe time box
        - Qualquer testador pode testar

■ Produtos de trabalho da execução do teste

- Documentação do status dos casos de teste individuais ou procedimentos
- Os relatórios de defeitos
- A documentação sobre os quais os itens de teste, objetos de teste, as ferramentas de teste e o testware estavam envolvidos no teste.
- Após a conclusão da execução o status de cada elemento da base pode ser determinado e relatado via rastreabilidade bidirecional.
- Permite verificar a cobertura

■ Produtos de trabalho de conclusão de teste

- Trabalho de conclusão incluem os relatórios de resumo de teste, os itens de ação para melhoria de projetos subsequentes ou iterações.

○ Rastreabilidade entre a base de teste e os produtos de trabalho de teste.

        - Produtos dos trabalhos de teste e os nomes desses produtos de trabalho variam de empresa pra empresa.
        - Importante manter a rastreabilidade durante todo o processo de teste entre cada elemento da base de teste.
            - Analisar o impacto das mudanças
            - Tornar o teste aditável
            - Atender aos critérios de governança de TI
            - Melhorar a compreensibilidade dos relatórios de progresso do teste e dos relatórios de resumo do teste.
            - Relacionar os aspectos técnicos do teste
            - Fornecer informações para avaliar a qualidade do produto.

Psicologia do teste

- Desenvolvimento de software, incluindo os testes de software, envolve seres humanos
- Identificação de defeitos pode ser percebida como uma crítica ao produto e ao seu autor.
- Viés de confirmação pode dificultar a aceitação das informações que não concordam com as crenças mantidas.
- Ser comunicado de maneira construtiva
    - Comece com colaboração em vez de batalha
    - Enfatize os benefícios do teste
    - Comunique os resultados dos testes e outras descobertas de uma maneira neutra.
    - Tente entender como a outra pessoa se sente e as razões pelas quais ela pode reagir negativamente
    - Confirme o que a outra pessoa entendeu e vice versa
- Mentalidade do testador e do desenvolvedor
    - A mentalidade de um testador deve incluir
        - curiosidade
        - pessimismo profissional
        - Olho critico
        - atenção aos detalhes
        - motivação para comunicação
        - relacionamento bons e positivos

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

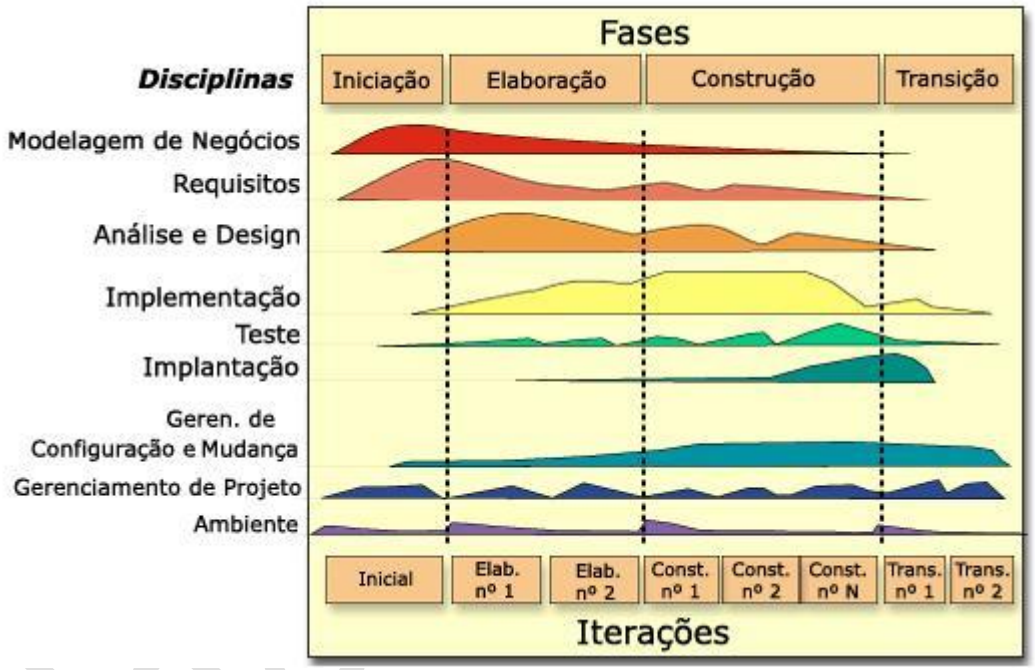
Capitulo 02 Teste durante o ciclo de vida de desenvolvimento de software

Modelos de ciclo de vida de desenvolvimento de software

- É uma forma de organizar o processo de desenvolvimento de software e como as atividades se relacionam com as outras de forma logica e cronológica.
- Diferença entre ágil e Tradicional
- Independentemente do modelo de ciclo de vida escolhido, as atividades de teste devem começar nos estágios iniciais
- Desenvolvimento de software e teste de software
  - Para cada atividade de desenvolvimento, existe uma atividade de teste correspondente. Ex: Criar diagrama X, tem que revisar diagrama X.
  - Cada nível de teste tem objetivos de teste específicos
  - A análise e modelagem de teste para um determinado nível de teste começam durante a atividade de desenvolvimento correspondente
  - Os testadores participam de discussões para definir e refinar os requisitos
  - Modelo mais comuns de desenvolvimento
    - Modelo sequencial. Ex: Cascata ou Waterfall
      - Fluxo sequencial e linear de atividade
      - Fase do processo de desenvolvimento deve começar quando a fase anterior estiver concluída.
      - Modelo cascata são concluídas uma após a outra
      - Modelo cascata as atividades de teste ocorrem somente após todas as outras atividades de desenvolvimento terem sido concluídas.
      - Modelo V é considerado sequencial
        - Modelo V: O processo de teste ao longo do processo, implementando o princípio testar no início. Níveis de teste associados a cada fase de desenvolvimento correspondente.
          - Levantamento, arquitetura, analise e desenvolvimento
          - Componente, integração, sistema e aceite
        - Modelo V: A execução dos testes associados a cada nível de teste ocorre sequencialmente, mas em alguns casos pode ocorrer a sobreposição
      - Exigem meses ou anos para serem entregues
    - Modelo iterativo e incremental. Ex: Scrum, Ágil, XP, Kanban
      - Envolve entender os requisitos, modelagem, construção e o teste.

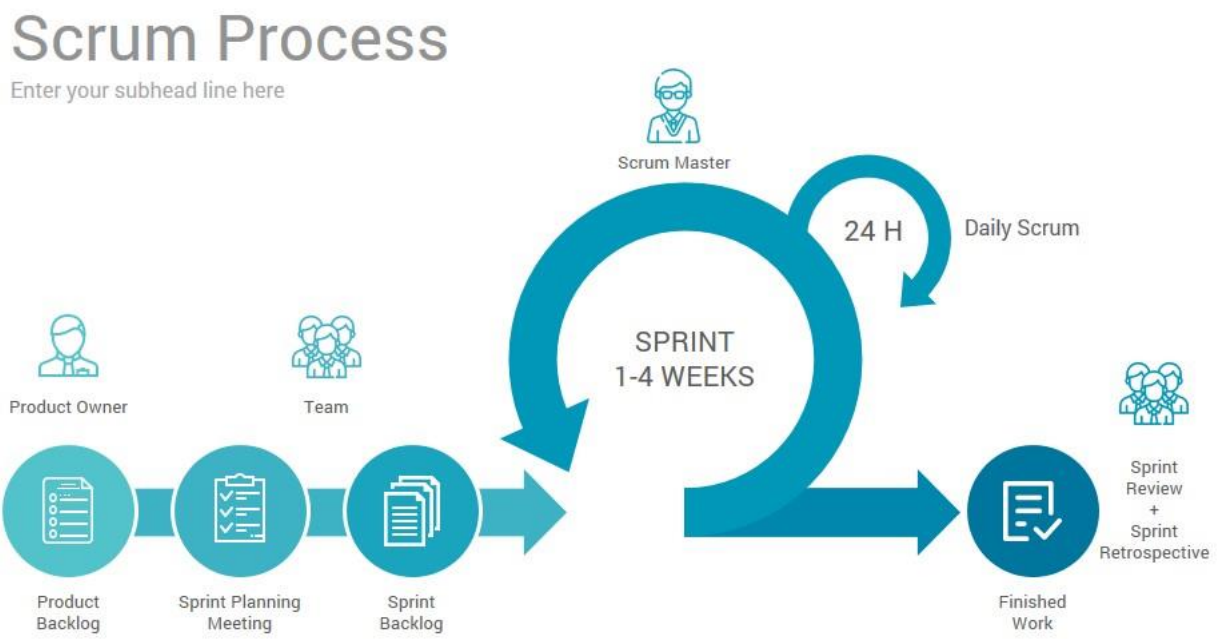
Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- O tamanho do incremento de recurso varia, com alguns métodos tendo partes maiores e algumas partes menores
- Desenvolvimento iterativo normalmente tem grupos de recursos são especificados, projetados, construídos juntos um serie de ciclo
- Desenvolvimento iterativo entrega ao cliente pequenos blocos do software
- Os recursos crescem de forma incremental, alguns métodos parte maiores e algumas parte menores.
    - RUP: Processo unificado da Racional: trabalha com iteração mais longas, entrega grupos de incrementos, não é ágil, mas é incremental
        - RUP

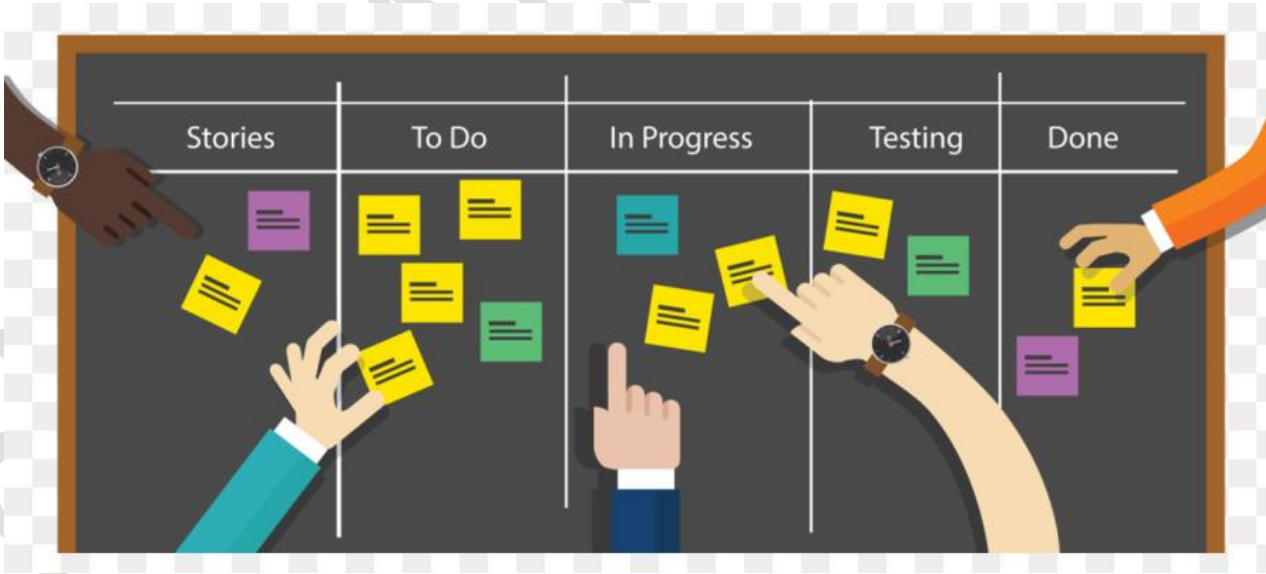


    - Scrum: Trabalha com iterações curtas, entrega partes de incrementos
        - Scrum

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com



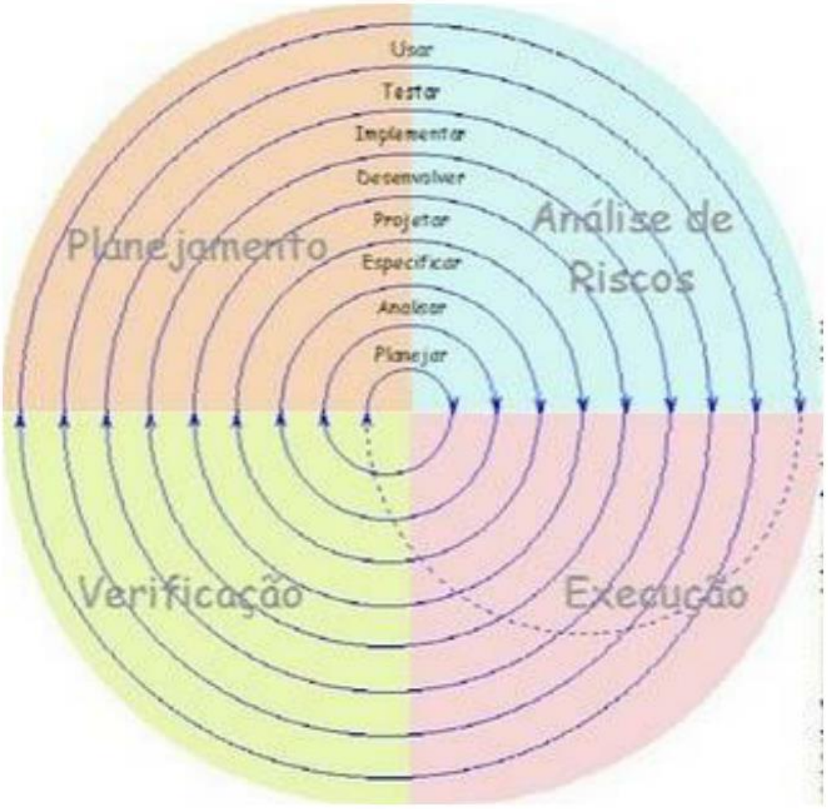
- Kanban: Trabalha com iterações curtas ou longas, entrega parte de incrementos
    - Kanban



- Espiral (prototipagem): Criação de incrementos experimentais. Planeja uma parte, realiza uma parte, check uma parte e libera uma parte.
    - Espiral

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

MODELO ESPIRAL



Voltar

- Componentes ou sistemas envolvem sobreposição e a interação dos níveis
- Ideal que cada componente seja testado em cada nível.
- Teste de regressão é cada vez mais importante à medida que o sistema cresce
- Esses métodos formam um sistema crescente, que pode ser liberado para os usuários finais em uma base de recurso, em uma iteração por iteração, ou em uma moda de liberação principal mais tradicional
- Em contraste com os modelos sequenciais, os modelos iterativos e incremental podem fornecer ao software utilizável em semanas ou até dias

- Os Modelos de ciclo de vida de desenvolvimento de software em contexto
  - Devem ser relacionados e adaptados ao contexto das características do projeto e do produto
  - Deve ser selecionado e adaptado com base na meta do projeto, no tipo do produto que está sendo desenvolvido, nas propriedades de negócios e nos riscos identificados.
  - Dependendo do contexto do projeto, pode ser necessário combinar ou reorganizar os níveis
  - Modelos de ciclo de vida podem ser combinados.
  - Motivos pelos quais os contextos podem ser adaptados.
    - Diferença nos riscos de produtos

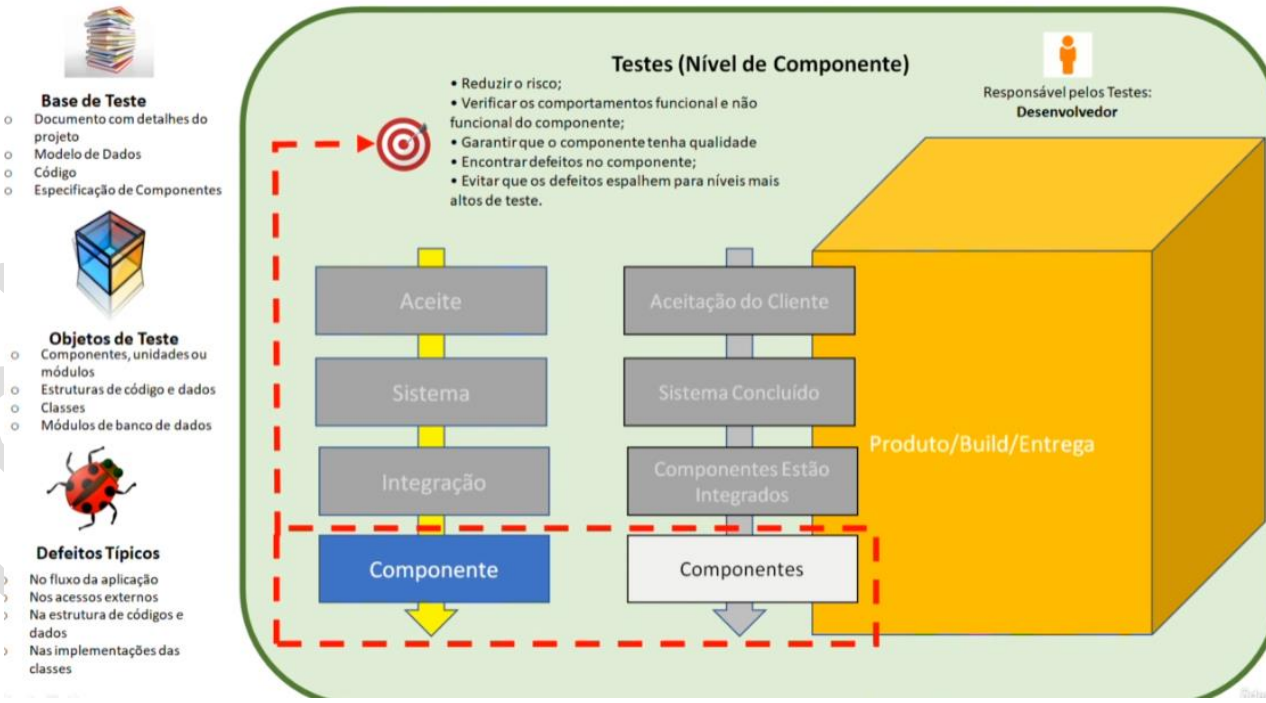
Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- Muitas unidades de negócios podem fazer parte de um projeto
- Pouco tempo para entregar um produto

Níveis de Teste

- Os Níveis de teste são grupos de atividades de teste que são organizados e gerenciados juntos
- Cada nível de teste é uma instância do processo de teste
- Nível de teste significa teste dinâmico
- Principais níveis de teste: Componente, Integração, Sistema e Aceite (CISA)
- Níveis de teste são caracterizados por:
  - Objetivos específicos
  - Base teste
  - Objeto de teste
  - Defeitos e falhas típicas
  - Abordagens e responsabilidades especificas
- Para cada nível de teste é necessário um ambiente de teste adequado
- Em alguns casos, especialmente no modelo ágil, os testes de regressão são automatizados
- (C) Teste de Componente:
  - Componentes que são testáveis separadamente
  - Podem exigir objetos simulados (Mock = Driver (Simula a chamada) ou Stubbing (Responder a chamada))
  - Teste funcional, características não funcionais e propriedade estruturais.
    - Reduzir riscos
    - Verificar comportamento funcional e não funcional do componente
    - Construir a confiança na qualidade componente
    - Encontrar defeitos no componente
    - Evitar que os defeitos espalhem para níveis mais altos
  - Base de teste:
    - Projeto detalhado
    - Código
    - Modelo de dados
    - Especificação de componentes

- Objeto de teste:
    - Componentes, unidade ou módulos
    - Estrutura de código e dados
    - Classes
    - Módulos de comunicação com o branco de dados.
- Defeitos típicos de falha:
    - Funcionalidade incorreta
    - Fluxo de dados
    - Código e logica incorretos.
- Geralmente é executado pelo desenvolvedor
    - Normalmente codificam e depois fazem o teste
    - Mas também podem escrever os testes automatizados de componentes antes da criação do programa (TDD = Test Driven Development (Desenvolvimento Orientado por testes)).
- Resumo



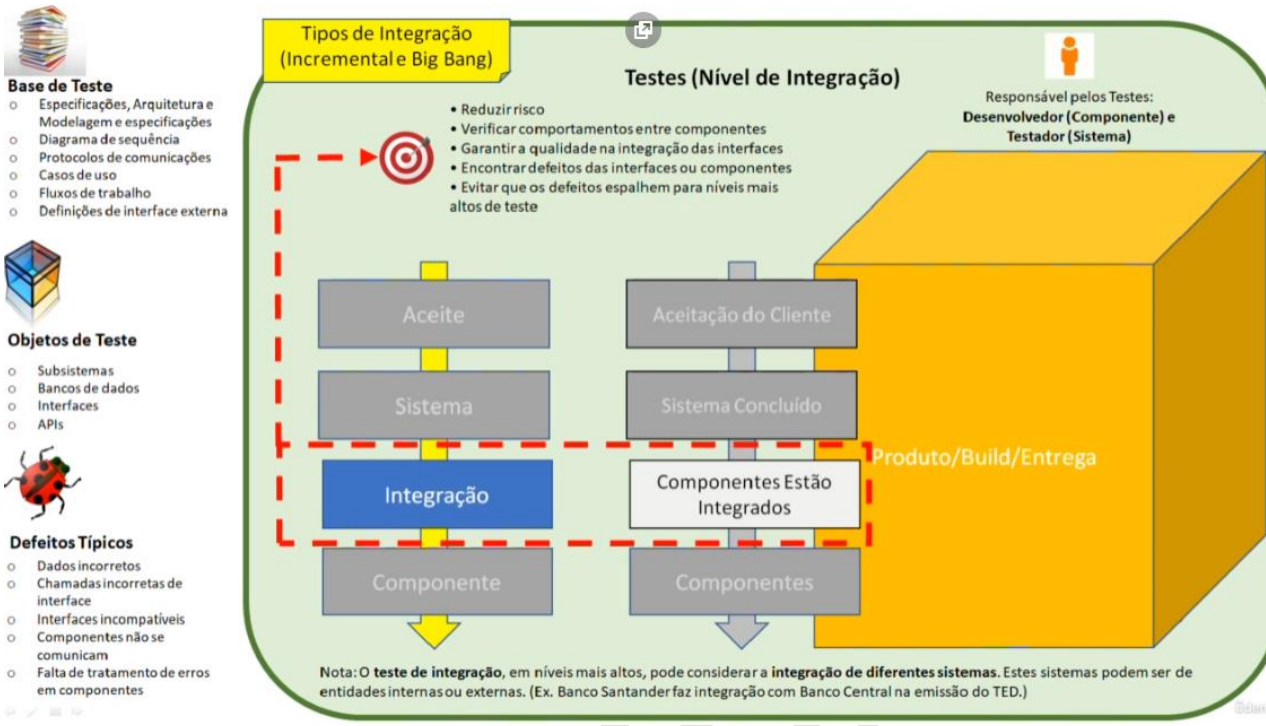
- (I)Teste de Integração (Teste de Serviço ou de API)
    - Teste entre interações entre componentes ou sistemas.
    - Reduzir risco
    - Verificar se os componentes funcionais e não funcionais das interfaces

- Construir confiança na qualidade das interfaces
- Encontrar defeitos
- Evitar que os defeitos espalhem para níveis mais altos
- Existem 2 níveis diferente:
    - Integração de componentes
        - Integrações entre componentes
        - Executado após o teste do componente e geralmente é automatizado.
        - Podem fazer parte do processo de integração contínua (É um processo que a cada build dispara uma bateria de teste de regressão)
        - Geralmente feito por desenvolvedores
    - Integração de sistemas
        - Interações entre sistemas pacotes e micro serviços
        - Não controla as interfaces externas.
        - O teste de integração de sistemas pode ser feito após o teste do sistema ou em paralelo com as atividades de teste do sistema em andamento.
        - Geralmente feito por Testadores
- Base de teste:
    - Software e modelagem do sistema
    - Diagrama de sequencia
    - Especificações de interface e protocolo de comunicação
    - Casos de uso
    - Arquitetura no nível de componente ou sistema
    - Fluxos de trabalho
    - Definições de interface externa
- Objetivos de teste:
    - Subsistemas
    - Banco de dados
    - Infraestrutura
    - Interfaces
    - APIs

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

        - Micro serviços
    - Defeitos típicos e falhas componentes e sistemas:
        - Dados incorretos, dados ausentes ou codificação incorreta.
        - Sequenciamento ou temporização
        - Incompatibilidade de interface
        - Falhas na comunicação entre componentes
        - Falha de comunicação não manipulada ou tratada de forma errada entre componentes
        - Suposições incorretas sobre o significado
        - Além desses acima o de sistema inclui:
            - Estruturas de mensagens inconsistentes entre sistemas
            - Falha de comunicação entre sistemas
            - Falha no cumprimento dos regulamentos de segurança obrigatórios
    - Os testes de integração devem se concentrar na própria integração
    - Devem ser feitos incremental (Pequeno número de componentes adicionais ou sistemas por vez.)
    - Podem ter teste estrutural, que leva em consideração a logica, estrutura do código
    - Quanto maior o escopo da integração. mais difícil se torna isolar os defeitos em um componente ou sistema específico
        - Esse é um dos motivos pelos quais a integração continua, onde o software é integrado componente por componente, geralmente inclui teste de regressão automatizado
    - Resumo

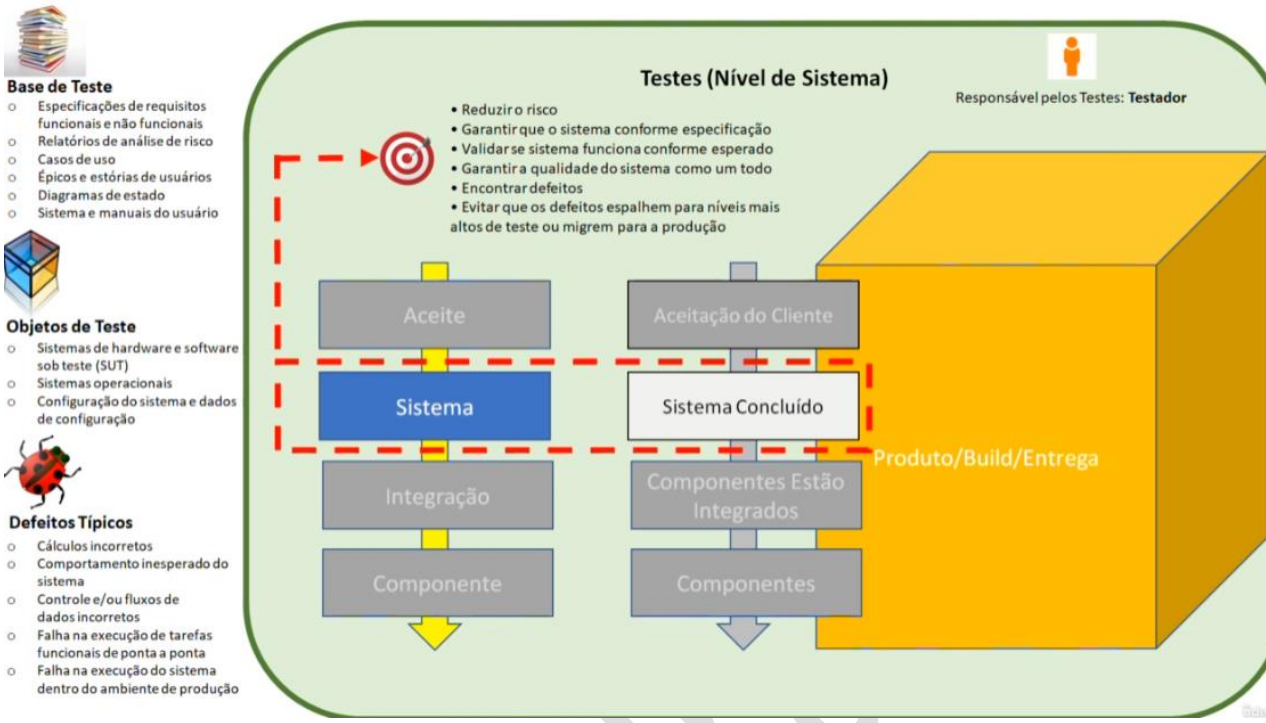
Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com



- (S)Teste de sistemas
  - Concentra no comportamento e nas capacidades de todo um sistema ou produto
  - Produz informações que são usadas pelas partes interessadas
  - Deve ser testado em um ambiente muito próximo de produção
  - Responsabilidade dos testadores de preferência independentes
  - Teste deve focar no comportamento de ponta a ponta
  - Execuções de ponta a ponta do sistema e os comportamentos não funcionais
    - Reduzir os riscos
    - Verificar se os componentes funcionais e não funcionais do sistema estão como projetados e especificados
    - Validar se o sistema está completo e funcionara como esperado
    - Criar confiança na qualidade
    - Encontrar defeitos
    - Evitar que os defeitos se espalhem
  - Base de teste:
    - Especificações de requisitos e sistemas
    - Relatório de análise de risco
    - Casos de uso
    - Épicos e histórias de usuário

        - Modelos comportamento do sistema
        - Diagrama de estado
        - Sistemas e manuais de usuário.
    - Objetivos do teste:
        - Aplicações
        - Sistemas de hardware e software
        - Sistemas operacionais
        - Sistema sob teste (SUT)
        - Configuração de sistema e dados
    - Defeitos e falhas:
        - Cálculos incorretos
        - Comportamento funcional e não funcional do sistema incorreto ou inesperado
        - Controle e fluxo de dados dentro do sistema
        - Falha na execução de tarefas
        - Falha no ambiente
        - Falha do sistema para funcionar conforme descrito nos manuais do sistema e do usuário.
    - Responsabilidade da equipe de teste
        - De preferência de testadores independentes
    - Resumo

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com



- (A)Teste de Aceite ou homologação
  - Se concentra no comportamento e na capacidade de todo um sistema ou produto
    - Estabelecer confiança na qualidade do sistema como um todo
    - Validar que o sistema está completo e funcionara como esperado
    - Verificar se os componentes funcionais ou não do sistema estão especificados.
  - Produz informação para avaliar a situação do sistema para implantação e uso pelo cliente
  - Formas comuns de teste de aceite:
    - Aceite do usuário (UAT)
      - Focado em validar a adequação do uso do sistema pelo usuário pretendido em um ambiente operacional real ou simulado.
      - O objetivo principal é desenvolver a confiança de que os usuários podem usar o sistema
    - Aceite operacional (OAT)
      - O teste de aceite do sistema pelas operações ou pela equipe de administração de sistemas geralmente é realizado em um ambiente de produção
      - Principal objetivo é criar confiança de que os operadores ou administradores do sistema possam manter o sistema funcionando adequadamente para os usuários.
        - Teste de backup e restauração
        - Instalar, desinstalar e atualizar

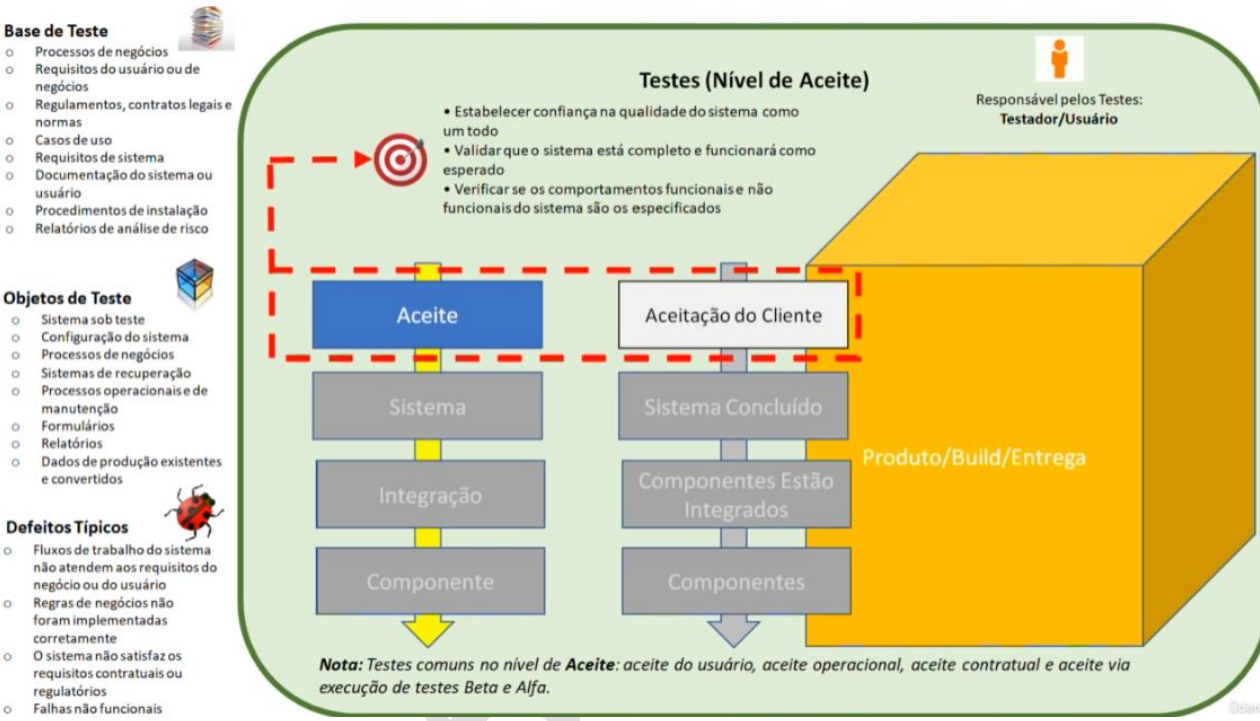
            - Recuperação de desastres
            - Gerenciamento de usuários
            - Tarefas de manutenção
            - Carregamento de dados e tarefas de migração
            - Verifica a vulnerabilidade
            - Teste de performance
    - Aceite contratual e regulatório
        - Realizado com base nos critérios de aceite de um contrato para desenvolver softwares específicos.
        - Realizado por testadores ou usuários
        - Principal objetivo é a criação de confiança de que a conformidade contratual ou regulatória foi alcançada
    - Alfa e beta
        - Realizado pra software lançado em grande escala, comercial ou prateleira
        - Objetivo é construção da confiança entre os clientes ou operadores
        - Detecção de defeitos relacionados as condições e ambientes em que o sistema será utilizado
        - Realizado no site da organização em desenvolvimento, não pela equipe de desenvolvimento, mas por clientes em potencial.
        - Teste alfa:
            - Tem a infraestrutura do fornecedor (Software e hardware)
            - Teste mais de logica do programa
            - Ambiente controlado
        - Teste Beta:
            - Software do fornecedor, mas o hardware é do cliente ou usuário.
            - Pode ocorrer após o teste alfa ou sem que este seja realizado
            - Problemas de incompatibilidade ou portabilidade.
- Base de teste:
    - Processos de negócios
    - Requisitos de usuário ou de negócios

- Regulamentos, contratos legais e normas
- Casos de uso
- Requisitos de sistemas
- Documentação do sistema ou usuário
- Procedimento de instalação
- Relatórios de análise de teste
- Procedimentos de backup e restauração, recuperação
- Requisitos funcionais
- Documentação de operações

- Objetivos de teste típicos
  - Sistema sob teste (SUT)
  - Configuração do sistema e dados de configuração
  - Processos de negócios para um sistema totalmente integrado
  - Sistema de recuperação e hot sites (Continuidade de negócios e testes de recuperação de desastres)
  - Processos operacionais
  - Formulários
  - Relatórios
  - Dados de produção existentes e convertidos
- Defeitos e típicos e falhas
  - Fluxos de trabalho do sistema não atendem aos requisitos do negocio
  - Regras de negócio não são implementados corretamente
  - Sistema não satisfaz os requisitos contratuais ou regulatórios
  - Falha não funcionais, como vulnerabilidade de segurança, eficiência de performance inadequada sob altas cargas ou operação inadequada em lima plataforma
- Abordagens e responsabilidades
  - Geralmente é responsabilidade dos clientes, usuários de negócios, proprietários de produtos ou operadores de um sistema
  - Teste de aceite é o último nível de teste em um ciclo de desenvolvimento
  - Teste de aceite de um produto de software de prateleiras (COTS) pode ocorrer instalado ou integrado

- Teste de aceite de um novo aprimoramento funcional pode ocorrer antes do teste do sistema
- Teste alfa e beta pode ocorrer no final de cada iteração

- Pode ter teste tipo caixa branca
- Resumo

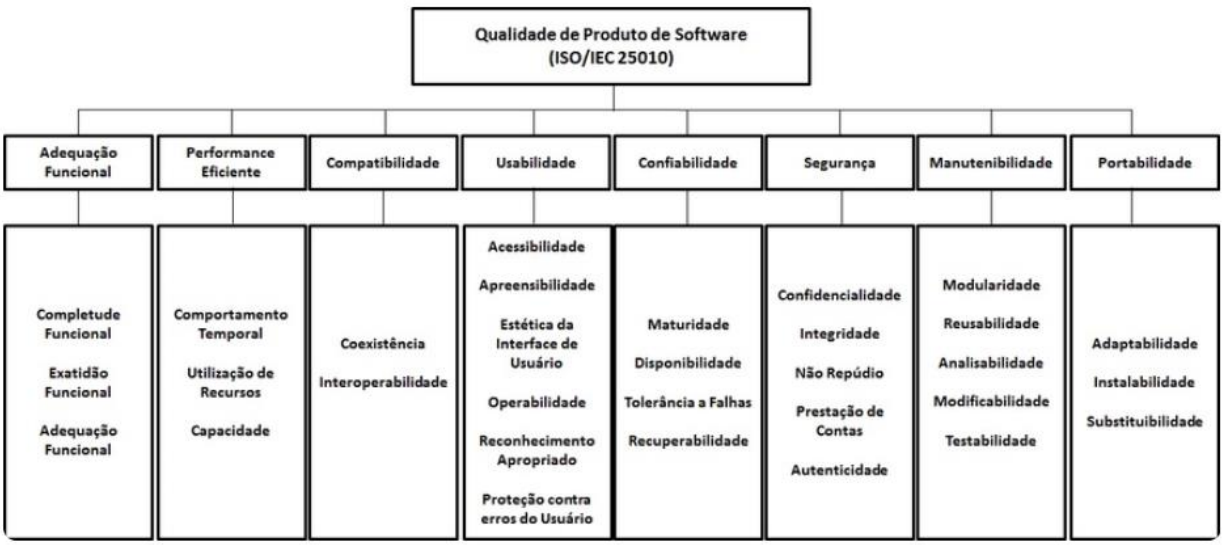


Tipos de Teste

- Pode ser realizado em qualquer nível de teste
- Grupo de atividades de teste destinado a testar características especificas de um sistema de software, ou parte de um sistema, com base em objetivos de teste específicos:
  - Avaliar as características de qualidade funcional, integridade, correção e adequação
  - Avaliar as características de qualidade não funcionais, confiabilidade, eficiência de performance, segurança, compatibilidade e usabilidade
  - Avaliar a estrutura ou arquitetura do componente ou sistema está correta
  - Avaliar os efeitos das alterações
  - Possível executar qualquer um dos tipos de testes citados abaixo em qualquer nível de teste
  - Teste de confirmação
    - Avaliar os efeitos das alterações, como a confirmação da correção de defeitos
  - Teste de regressão
    - Procurar as alterações não intencionais no comportamento como o resultado de alterações no software ou no ambiente

- Teste caixa preta
    - Teste funcional:
        - Avaliam as funções que o sistema deve executar
        - As funções são “o que” o sistema deve fazer
        - Devem ser realizados em todos os níveis de teste
        - Pode ser medido através da cobertura funcional
        - Pode envolver habilidades ou conhecimentos especiais.
        - Começa a partir do teste de componente
    - Teste não funcional:
        - Avaliam as características de sistemas e de software, como usabilidade, eficiência de performance e ou segurança.
        - ISO-25010
        - Teste não funcional é o teste de “quão bem” o sistema se comporta
        - Pode e geralmente deve ser realizado em todos os níveis de teste e feito o mais cedo possível.
        - Pode ser medido por meio de cobertura não funcional
        - Pode envolver habilidades ou conhecimentos especiais
        - UPS (Usabilidade, performance e segurança) 3 principais testes não funcionais
        - Começa a partir do teste de componente
- Teste caixa branca:
    - Com base na estrutura interna ou na implementação do sistema
    - Pode ser medido através da cobertura elemento estrutural.
    - Nível de teste de componente são chamados de Cobertura de código
        - Baseada na porcentagem do código do componente que foi testado.
    - Nível de teste de integração de componentes são chamados de cobertura estrutural
        - Porcentagem de interfaces exercidas pelos testes.
    - Pode envolver habilidades ou conhecimentos especiais.
    - Normalmente é feita pelo desenvolvedor
- Teste Relacionados a mudanças:
    - Quando são feitas mudanças em um sistema deve se testar para confirmar que as alterações ou defeitos foram corrigidos.

- Podem ocorrer em desenvolvimento interativo e incremental
    - Teste de confirmação (Local ou Reteste)
        - Finalidade desse teste é confirmar se o defeito original foi corrigido com sucesso.
        - Pode ser realizado em todos os níveis
        - Retestar um único componente
    - Teste de regressão (Global)
        - Possível que alguma alteração no código possa afetar acidentalmente o comportamento de outras partes do sistema
        - Pode ser realizado em todos os níveis
        - Finalidade desse teste envolve a execução de testes para detectar esses efeitos colaterais indesejados
        - São executados muitas vezes
        - Testar todo o sistema
        - Forte candidato a automação
- Exemplos de Tipos e níveis de teste
    - 8 características e 33 subcaracterísticas



- Resumão
    - Tipo de teste é uma forma de testar existem 3 tipos de testes: Teste funcionais, não funcionais e relacionado a mudanças. Na Iso 25010 existem 8 características e 33 sub características. Adequação funcional e funcionalidade significam a mesma coisa, são testes relacionados a negócio As outras características são testes técnicos ou não funcionais Os tipos de testes técnicos não funcionais mais comuns são (UPS) usabilidade, performace e segurança Teste caixa preta, executar o software ou

comportamento, tem a divisão funcional ou não funcional Teste caixa branca, vai avaliar a lógica ou código fonte (Teste componente, teste integração de componente) Teste caixa preta e teste de caixa branca se aplica em todos os níveis Teste baseado na experiência, onde o testador tem que ter conhecimento similar, ramo de negócio ou conhecimento naquela tecnologia Níveis de teste são as etapas de teste dinâmico CISA (Componente, integração, sistema e aceite) ou CISIA (Componente, integração de componente, sistema, integração de sistemas e aceite) Existem teste funcional e não funcional em todos os níveis

Teste de manutenção:

- Após serem implantados em ambientes de produção, o software e os sistemas precisam ser mantidos
- Vários tipos de mudanças são quase inevitáveis em software e sistemas:
  - Corrigir defeitos
  - Adicionar nova funcionalidades
  - Remover ou alterar funcionalidade entregues.
- Manutenção é necessária para preservar ou melhorar as características de qualidade não funcionais do componente ou do sistema ao longo de sua vida
- Teste de manutenção se concentra em testar as alterações no sistema, bem como em testar as partes inalteradas que podem ter sido afetados pelas alterações.
- Escopo depende:
  - Grau de risco da mudança
  - Tamanho do sistema existente
  - Tamanho da mudança
- Gatilho para manutenção (Hot Fixes)
  - Motivos para começar o teste de manutenção ocorre alguma mudança:
    - Modificação: Mudança, correção, melhoria em algum componente do sistema, funcionalidade
    - Migração: Como de uma plataforma para outra, comparar os sistemas pra conhecer as funcionalidades com o novo sistema.
      - Precisa de uma garantia que se acontecer algum problema, tenha uma forma de voltar atrás.
    - Aposentadoria: Quando o aplicativo atinge o fim de sua vida
    - Quando um aplicativo ou sistema é retirado, isso pode exigir o teste de migração ou arquivamento de dados.
    - Garantir antes da modificação, migração ou aposentadoria o procedimento de restauração.

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- A análise do impacto avalia as alterações feitas para uma liberação de manutenção identificando os resultados esperados, os possíveis efeito colaterais provocado e para identificar as áreas dos sistemas que serão afetadas.
- A análise de impacto pode ser difícil se:
    - Especificações estão desatualizadas ou ausentes
    - Casos de teste não estão documentados ou estão desatualizados
    - A rastreabilidade bidirecional entre os testes e a base de teste não foi mantida
    - O suporte das ferramentas é fraco ou inexistente
    - As pessoas envolvidas não possuem conhecimento do domínio ou sistema
    - Não foi dada atenção suficiente a manutenção do software durante seu desenvolvimento

Capitulo 03 Teste estático

Noções básicas sobre testes estáticos:

- Em contraste com o teste dinâmico o Teste Estáticos envolve em ler, comparar...
- Teste estático depende do exame manual dos produtos de trabalho, isto é, revisões
    - Revisão
- Teste estático depende da avaliação orientada por ferramentas do código ou produtos de trabalho, isto é, análise estática, feita automática.
    - Análise estática
- Os dois tipos acima avaliam o código do produto sem executar o código.
- Analise estática é importante para sistemas críticos de segurança e frequentemente incorporada aos sistemas automatizados de criação e distribuição.
- Ramo de A análise estática também é frequentemente incorporada aos sistemas automatizadas de criação e distribuição, por exemplo, no desenvolvimento ágil, na entrega contínua e integração contínua
- Produtos de trabalho que podem ser examinados por testes estáticos
    - Produtos que podem ser examinados por testes estáticos:
        - Especificações, incluindo requisitos de negócios, requisitos funcionais e requisitos de segurança.
        - Épicos, histórias de usuários e critérios de aceite
        - Especificações de arquitetura e design
        - Código
        - Testware

        - Guias de usuários
        - Páginas Web
        - Contratos, planos de projetos, cronograma e orçamentos
        - Diagrama de atividades que podem ser usados para testes baseados em modelo.
    - As avaliações podem ser aplicadas a qualquer produto de trabalho que os participantes saibam ler e entender.
    - Analise estática pode ser aplicada em uma estrutura formal (Normalmente código ou modelos) como também aplicada na linguagem natural (verificação de ortografia, gramatica...)
- Benefícios do teste estático:
    - Quando aplicado no início permite a detecção antecipada dos defeitos antes da realização dos testes dinâmicos
    - Localizar e corrigir os defeitos rapidamente é quase sempre muito mais barato para a organização
    - Detectar e corrigir defeitos com mais eficiência e antes da execução dinâmica
    - Identificar os defeitos que não são facilmente encontrados por testes dinâmicos
    - Prevenir os defeitos no desenho ou codificação
        - Para o ISTQB toda vez que um defeito é encontrado antes do teste dinâmico é uma prevenção
    - Aumentar a produtividade no desenvolvimento
    - Reduzir os custos do tempo de desenvolvimento, custo e tempo do teste, custo total de qualidade durante a vida útil do software
    - Melhorar a comunicação entre os membros da equipe
- Diferenças entre estático e dinâmico:
    - Estático sistema não está em execução! = Dinâmico sistema está em execução
    - Estático encontra-se defeito no dinâmico encontra-se falha
    - Estáticos e dinâmicos tem o mesmo objetivo
    - Estáticos e dinâmicos se completam
    - Encontra defeitos diretamente em produtos de trabalho, em vez de identificar as falhas causadas por defeitos quando o software é executado.
    - Se o caminho onde o defeito está difícil de alcançar ou de ser testado, então não será fácil construir e executar um teste dinâmico.

- Teste estático pode ser usado pra melhorar a consistência e a qualidade interna dos produtos de trabalho, enquanto o teste dinâmico geralmente se concentra em comportamentos externamente visíveis
- Defeitos comuns testes estáticos:
    - Defeitos em requisitos (Ambiguidades, contradições)
    - desenho (Algoritmos ineficientes ou estruturas banco de dados.)
    - codificação (Variáveis com valores indefinidos, variáveis declaradas...)
    - Desvios de padrões (Falta de aderência aos padrões de codificação)
    - especificações incorretas na interface (Unidades de medidas diferentes usadas pelo sistema)
    - vulnerabilidade segurança (sensibilidade a estouro do buffer)
    - Lacunas ou imprecisões na rastreabilidade (Falta de teste para um critério de aceite).
- Maioria dos defeitos de manutenção só podem ser encontrados por teste estáticos

Processo de revisão

- As revisões variam de formal para informal
- Informais caracterizam por não seguir um processo definido e não ter um resultado formal documentado
- Formais caracterizam pela participação da equipe, com resultados documentados e procedimentos documentos para revisão. A formalidade depende do modelo de ciclo de vida, maturidade, complexidade do produto...
- O foco da revisão depende dos objetivos acordados da revisão
- ISO20246 contém descrições mais detalhadas do processo de revisão
- Processo de revisão compreende as seguintes atividades (PIRAC)
    - Planejar
        - Definir o escopo, que inclui o propósito da revisão, quais documentos ou partes de documentos que devem ser revisados
        - Estimativa de esforço
        - Identificar as características da revisão
        - Selecionar as pessoas para participar da revisão
        - Definir os critérios de entrada e saída para os tipos de revisão mais formais
        - Verificar se os critérios de entrada foram atendidos
    - Iniciar Revisão:

        - Distribuir o produto de trabalho e outros materiais
        - Explicar o escopo, os objetivos, o processo...
        - Responder a quaisquer perguntas que os participantes possam ter sobre a revisão
    - Revisão individual:
        - Rever todo ou parte do produto de trabalho
        - Observar os possíveis defeitos
    - Analisar e comunicar:
        - Comunicar possíveis defeitos identificados
        - Analisar possíveis defeitos, atribuindo responsáveis
        - Avaliar e documentar características e qualidades
        - Avaliar as conclusões da revisão em relação aos critérios de saída para tomar decisões de revisão
    - Corrigir e reportar:
        - Criar relatórios de defeitos para as descobertas que exigem mudança
        - Corrigir os defeitos encontrados
        - Comunicar os defeitos a pessoa da equipe
        - Registrar o status atualizado dos defeitos
        - Métricas de coleta
        - Verificar se os critérios de saída foram estabelecidos
        - Aceitar os produtos de trabalho quando os critérios de saída são atingidos
- Funções e atividades de uma revisão formal:
    - Autor:
        - Criar o produto de trabalho
        - Corrigir os defeitos no produto de trabalho sob revisão
    - Gestor
        - Responsável pelo planejamento da revisão
        - Decidir sobre a execução das revisões
        - Atribuir pessoal, orçamento e tempo
        - Monitorar a rentabilidade continua

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- Facilitador (Moderador):
    - Garantir a execução eficaz das reuniões de revisão
    - Mediar, se necessário entre vários pontos de vista
    - Frequentemente a pessoa sobre quem o sucesso da revisão depende
- Lider de revisão:
    - Assumir a responsabilidade geral pela revisão
    - Decidir quem será envolvido organizar quando e onde acontecerá a revisão
- Revisor:
    - Podem ser especialista no tema, pessoas trabalhando no projeto
    - Identificar possíveis defeitos no produto de trabalho
    - Pode representar diferentes perspectivas
- Redator (Registrador):
    - Coletar possíveis defeitos encontrados durante a atividade de revisão
    - Registar novos defeitos potenciais, pontos em aberto e decisões da reunião de revisão
- Em alguns tipos de revisão uma pessoa pode desempenhar mais de uma função

- Tipos de revisão
    - Principal objetivo da revisão é descobrir defeitos
    - Todos os tipos de revisões podem auxiliar na detecção de defeitos
    - Podem ter mais de um tipo de revisão por produto
    - Se mais de um tipo de revisão for usado, o pedido pode variar
    - Podem ser feitas como revisões em pares, ou seja, feitas por colegas em um nível organizacional aproximado ou semelhante
    - Quatro tipos mais comuns de revisões (IATI)
        - Revisão informal:
            - Objetivo principal é detectar defeitos potenciais
            - Outros objetivos seriam gerar novas ideias ou soluções, resolvendo rapidamente pequenos problemas
            - Não baseado em um processo formal
            - Não pode envolver uma reunião de revisão
            - Pode ser realizado por um colega do autor

    - Os resultados podem ser documentados
    - Varia de utilidades dependendo dos revisores
    - Checklist é opcional
    - Muito usado no desenvolvimento ágil
- Acompanhamento:
    - Principais defeitos encontrar defeitos, melhorar o produto
    - Outros objetivos trocam de ideias sobre técnicas ou variações de estilo
    - Preparação individual antes da reunião é opcional
    - A reunião normalmente é feita pelo autor do produto
    - Redator é obrigatório
    - Checklist é opcional
    - Pode assumir formas de cenários, teste práticos ou simulações
    - Possíveis log de defeitos e relatórios de revisão podem ser produzidos
    - Pode variar na pratica de bastante informal para muito formal
- Revisão técnica:
    - Principal objetivo, obtenção de consenso e a detecção de defeitos comuns
    - Outros objetivos avaliar a qualidade e criar confiança do produto
    - Revisor deve ser par técnicos do autor
    - Necessário preparação individual antes da reunião
    - Reunião de revisão é opcional
    - Redator é obrigatório e preferencialmente que não seja o autor
    - Checklist é opcional
    - São produzidos registros de defeitos potenciais e o relatório de revisão
- Inspeção:
    - Principais defeitos encontrar defeitos potenciais, avaliar a qualidade e criar confiança no produto
    - Outros objetivos, motivar e capacitar os autores a melhorar futuros produtos
    - Segue um processo definido com saídas documentadas formais, com base em regras e checklist

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

        - Utiliza funções claramente definidas
        - Necessário preparação individual antes da reunião
        - Revisores são pares do autor ou especialista
        - São usados critérios de entradas e saídas
        - Redator é obrigatório
        - Reunião de revisão é liderada por um facilitador treinado
        - O autor não pode atuar como líder de revisão, leitor ou redator
        - São produzidos registros de defeitos potenciais e o relatório de revisão
        - As métricas são coletadas e usadas para melhorar todo o processo.
    - Revisão pode revisar um único documento ou vários documentos
- Aplicando técnicas de revisão (AC2P2)
    - Ad Hoc:
        - Os revisores recebem pouca ou nenhuma orientação sobre como essa tarefa deve ser executada
        - Os revisores geralmente leem o produto de trabalho, identificam e documentam os problemas à medida que o encontram.
        - É uma técnica comumente usada que requer pouca preparação
        - Altamente dependente das habilidades do revisor
        - Pode levar a muitos problemas relatados em duplicidades se forem feitos por revisores diferentes
    - Baseada em Checklists:
        - Principal vantagem é uma técnica sistemática, onde os revisores detectam os problemas com base em checklist que são distribuídos no início da revisão
        - Consiste um conjunto de perguntas baseadas em possíveis defeitos, que podem ser derivados da experiencia
        - Devem ser específicos para o tipo de produtos e devem ser mantidos regularmente para cobrir os tipos de problemas perdidos nas revisões anteriores
        - Deve-se tomar cuidado para não seguir apenas o checklists na revisão individual, mas também procurar defeitos fora do checklists.
    - Baseada em Cenários (teste práticos):
        - Revisores recebem orientação estruturadas sobre como ler o produto de trabalho.

- Suporte aos revisores na execução de sequencias no produto com base no uso esperado do produto de trabalho, se o produto for documentado como caso de uso
- Fornecem melhores diretrizes sobre como identificar tipos específicos de defeitos do que simples entradas no checklist
- Os revisores não devem ser restritos aos cenários documentados.

■ Baseada em Perspectiva:

- Semelhante a baseada em papeis
  - Se estiver falando em olhar o software na visão do cliente.
- Revisores assumem pontos de vista de diferentes Stakholders (Pontos de vista incluem Usuário final, marketing, designer, testador ou operador) na revisão individual.
- Leva a uma maior profundidade na revisão individual, com menos duplicidade de problemas entre os revisores
- Revisores tentem usar o produto de trabalho sob revisão para gerar o produto que eles obteriam
- Espera-se serem usados checklist
- É mais eficaz para revisar os requisitos e os produtos de trabalho técnico.

■ Baseada em papeis:

- Se estiver falando em olhar o software na visão de perfil do usuário final
- Avaliam o produto de trabalho na perspectiva dos papeis individuais dos stakeholders
- Os papeis incluem tipos específicos de usuário final e papeis específicos na organização
- Pode ser usado checklist

○ Fatores de sucesso para revisão

■ Fatores organizacionais de sucesso para revisões:

- Cada revisão tem objetivos claros, definidos durante o seu planejamento e usados como critérios mensuráveis de saída.
- São adequados para alcançar os objetivos e são apropriados ao tipo e ao nível de produtos de trabalho de software e participantes
- Quaisquer técnicas são adequadas para a identificação de defeitos no trabalho revisado
- Todos os checklists abordam os principais ricos e deverão estar atualizados

- Documentos grandes são escritos e revisados em pequenos pedaços, para que os feedbacks antecipados
- Os participantes devem ter tempo suficiente para se preparar
- As revisões são agendadas com aviso adequado
- A gerencia apoia o processo de revisão

- Fatores de sucesso relacionados as pessoas:
  - As pessoas certas estão envolvidas para atender aos objetivos
  - Os testadores são vistos como revisores valiosos que contribuem para revisão
  - Os participantes dedicam tempo e atenção adequados aos detalhes
  - As revisões são realizadas em pequenos trechos
  - Os defeitos encontrados são reconhecidos, apreciados e manipulados objetivamente
  - A reunião é bem administrada
  - A revisão é conduzida em um ambiente de confiança
  - Os participantes devem evitar linguagem corporal e comportamento que possam indicar tedio
  - O treinamento adequado é fornecido
  - Uma cultura de aprendizado e melhoria de processo é promovida

Capitulo 04 Técnicas de teste

Categorias de técnicas de teste

- Quantos teste e quais teste tenho que fazer
- A escolha de quais das técnicas de teste usar depende de vários fatores:
  - Complexidade do componente ou do sistema
  - Normas Regulatórias
  - Requisitos contratuais ou do cliente
  - Níveis e tipo de risco
  - Documentação disponível
  - Conhecimento e habilidades do testador
  - Ferramentas disponíveis
  - Tempo e orçamento

        - Modelo de ciclo de vida do desenvolvimento de software
        - Tipos de defeitos esperados no componente ou sistema.
    - Algumas são aplicadas em determinadas situações de nível de teste ou em todos os níveis de teste.
    - Ao criar casos de teste, os testadores geralmente usam uma combinação de técnicas de teste para obter os melhores resultados do esforço de teste
    - O uso de técnicas pode variar de muito informal a muito formal, o nível adequado depende do contexto dos testes.
    - Categorias de técnicas de teste e suas características
        - As técnicas de testes são classificadas como:
            - Caixa preta ou comportamentais ou baseada no comportamento
                - São fundamentadas em uma análise da base de teste apropriada
                - São aplicadas a testes funcionais (Negocio) e não funcionais (Técnica)
                - Se concentram nas entradas e saídas do objeto de teste sem referência a sua estrutura interna
            - Caixa branca ou estruturais ou baseada na estrutura
                - São baseadas em análise da arquitetura, do detalhamento do projeto da estrutura interna ou o código do objeto
                - Concentram-se na estrutura e no processamento dentro do objeto de teste
            - Baseada na experiencia
                - Aproveitam o conhecimento dos testadores, desenvolvedores e usuário para projetar, implementar e executar teste
                - São combinadas com caixa preta e caixa branca.
        - Características comuns:
            - Caixa preta:
                - As condições de teste, os casos de teste e os dados são derivados de uma base de teste que pode incluir requisitos de software, especificações, casos de uso e história de usuário.
                - Casos de teste podem detectar lacunas entre os requisitos e as suas implementações, bem como desvios nos requisitos
                - Cobertura é medida com base nos itens testados na base de teste e na técnica aplicada nesta base.
            - Caixa Branca:

        - As condições de teste, os casos de teste e os dados são derivados de uma base de teste que pode incluir código, arquitetura de software, o detalhamento do projeto ou qualquer outra fonte de informação relacionada a estrutura do software

        - Cobertura é medida com base nos itens testados em uma estrutura selecionada, aplicada a base de teste

    - Baseada na experiencia:

        - As condições de teste, os casos de teste e os dados são derivados de uma base de teste pode incluir conhecimento e a experiencia de testadores, desenvolvedores, usuários e stakeholders

- ISO / IEC / IEEE 29119-4 (Técnicas de teste)

Teste Caixa preta

- Particionamento de equivalência

    - Testar os valores de cada grupo

    - Divide os dados em partições (Classes de equivalência), de tal forma que todos os membros de uma determinada partição deve ser processado da mesma maneira.

    - Tipos de partições:

        - Validos:

            - Valores aceitos pelo componente do sistema, chamada de partição de equivalência valida

        - Inválidos

            - Valores devem ser rejeitados pelo componente do sistema, chamada de partição de equivalência invalida

        - As partições podem ser identificadas para qualquer elemento de dados relacionados ao objeto de teste, incluindo entradas, saídas, valores internos, valores relacionados a tempo e para parâmetros de interface.

        - Se necessário, qualquer partição pode ser dividida em subpartições

        - Cada valor deve pertencer a uma partição de equivalência

        - Partições de equivalência invalida são usadas em casos de teste, elas devem ser testadas individualmente, ou seja, não podem ser combinadas com outras partições de equivalência invalida.

    - Cobertura de 100% os casos de teste devem cobrir todas as partições identificas, a cobertura é medida como o número de partições testadas por pelo menos um valor, dividido pelo número de partições de equivalência identificadas, normalmente expresso em porcentagem

    - Aplicável em todos os níveis de teste.

    - Exemplo

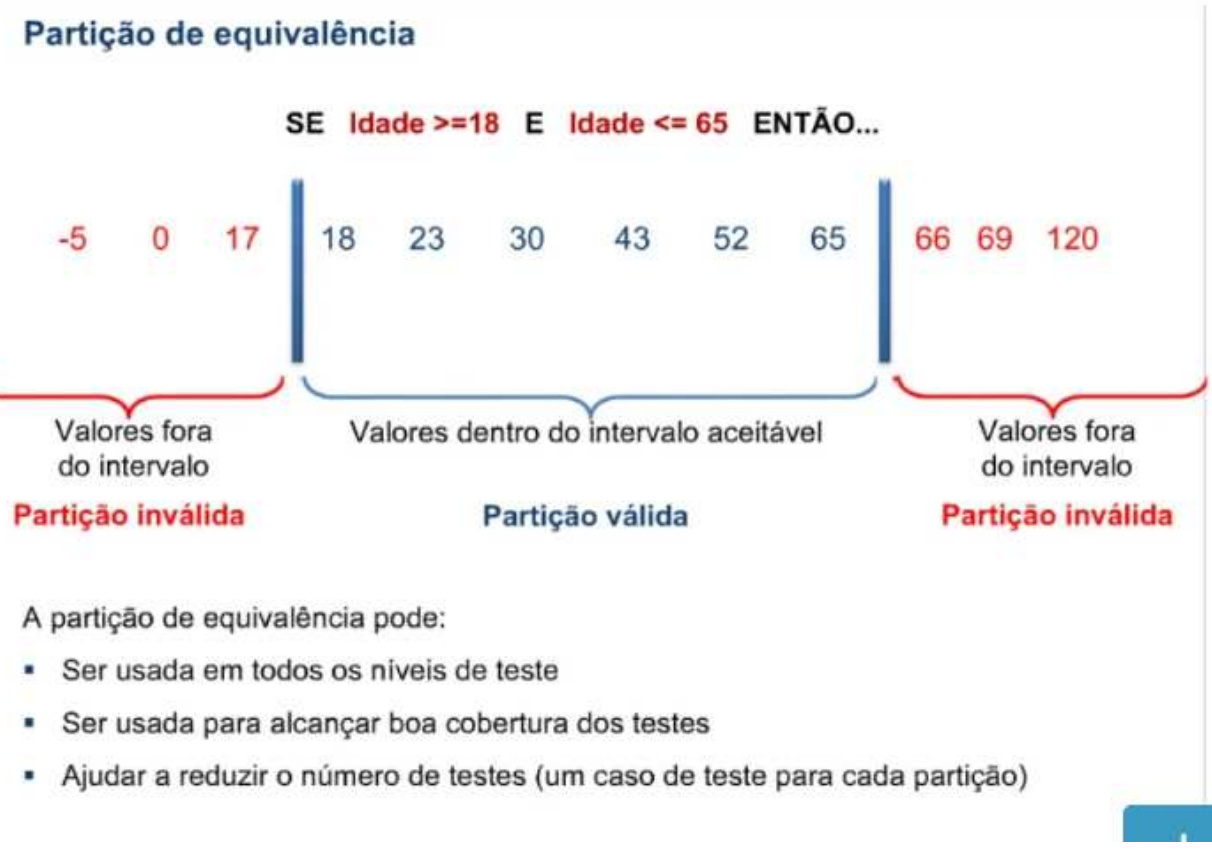
- Exemplo de Particionamento de equivalência

• Exemplo de aplicação das diretrizes:

a) Se uma condição de entrada especifica uma faixa de valores, uma classe de equivalência válida e duas inválidas são definidas

- Entrada: campo idade deve aceitar valores de 1 a 99 anos.
  - Faixa de valor: 1 a 99
  - Classe válida: números de 1 a 99
  - Classe inválida: números menores que 1
  - Classe inválida: números maiores que 99

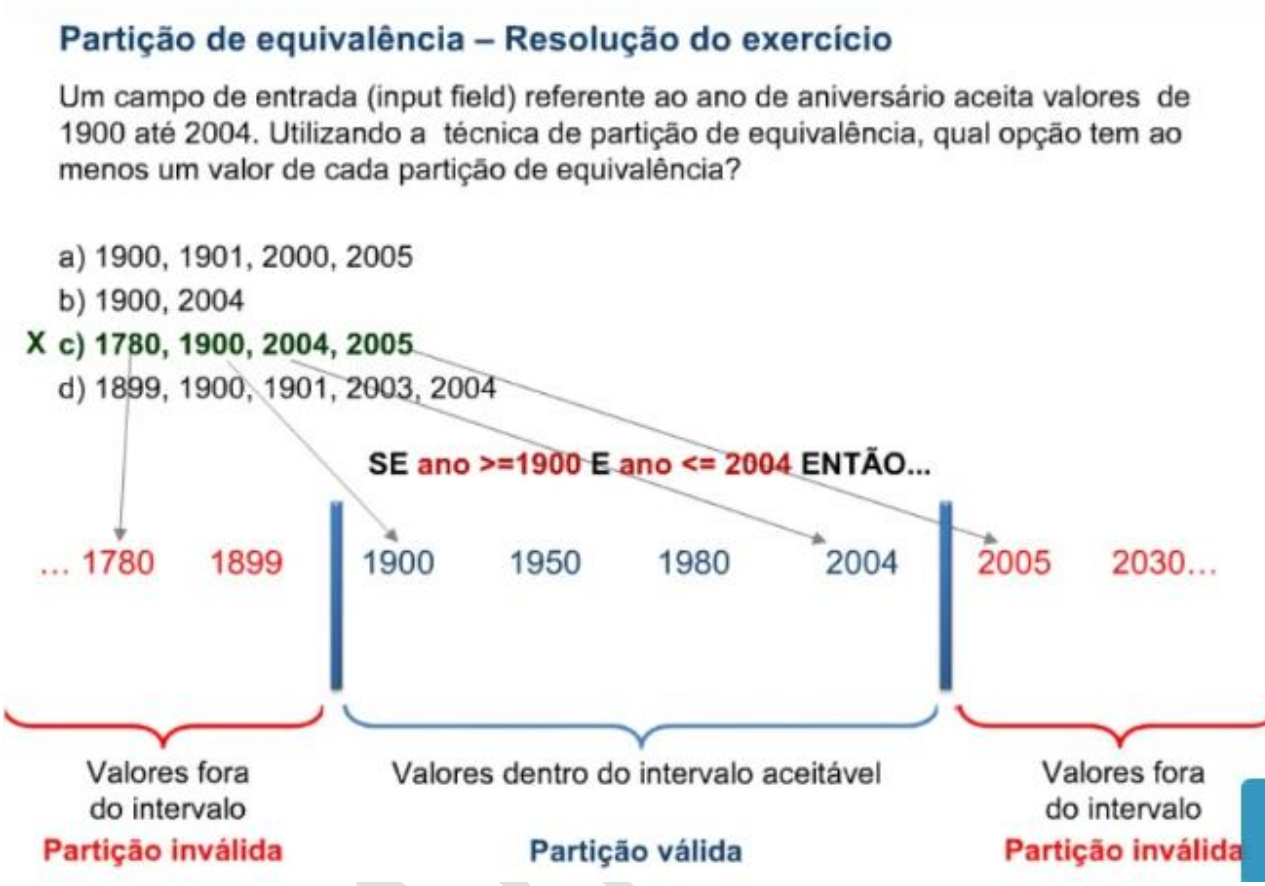
- Exemplo de Particionamento de equivalência



A partição de equivalência pode:

- Ser usada em todos os níveis de teste
- Ser usada para alcançar boa cobertura dos testes
- Ajudar a reduzir o número de testes (um caso de teste para cada partição)

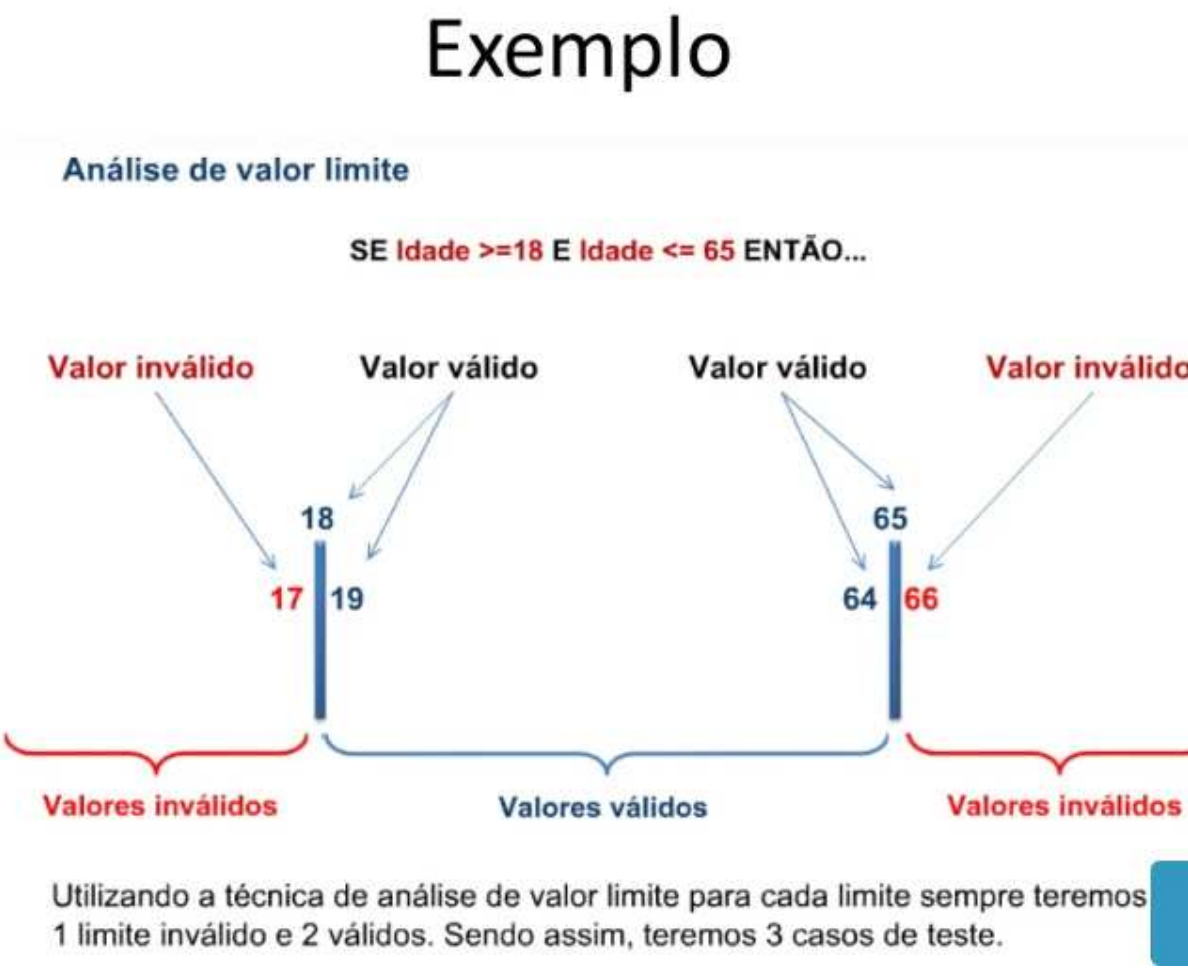
- Exemplo de Particionamento de equivalência



- Analise de valor limite
  - É uma extensão do particionamento de equivalência, mas só pode ser usada quando a partição é ordenada, constituindo dados numéricos e sequenciais.
  - Valores mínimo e máximo são os valores limites
  - Existem 3 partições de equivalência:
    - Invalido (Muito baixo), valido e invalido (Muito alto)
  - O comportamento nos limites das partições de equivalência é mais provável que seja incorreto do que o comportamento dentro das partições
  - Pode ser aplicada em todos os níveis de teste
  - Normalmente é usada para testar requisitos que exigem um intervalo de números
  - A Cobertura é medida como o número de valores limites testados, dividido pelo número total de valores de teste de limite identificado.
  - Exemplo
    - Exemplo Analise de valor limite

Estrada prevista	Saída esperada
19	Inválido
20	Válido
30	Válido
31	Inválido

- Exemplo Analise de valor limite



- Exemplo Analise de valor limite

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

Exemplo

- (CTFL - BSTQB) Um campo de entrada referente a data de nascimento aceita valores de 1860 até 2860. Utilizando a análise de valor limite o teste usaria quais valores.

a) 0, 1860, 2860, 3000
b) 1860, 2860
c) 1859, 1900, 1861, 2859, 2860, 2861
d) 1859, 1860, 2860, 2861

- Teste de tabela de decisão
  - Técnica de testes combinatórios
  - São uteis para testar diferente condições de combinação levam a resultados diferentes
  - Uma abordagem é o teste tabela de decisão
  - Ao criar tabela de decisão o testador identifica as condições e as ações resultantes do sistema.
  - Formam linhas da tabela, geralmente com as condições no topo e as ações na parte inferior.
  - Cada coluna corresponde a uma regra de decisão que define uma combinação exclusiva de condições que resultam na execução das ações associadas a regra.
  - Os valores são geralmente mostrados como valores booleano ou valores discretos, mas também podem ser números ou intervalos numéricos.
  - A notação comum nas tabelas de decisões é a seguinte:
    - Para condições:
      - "S" Condição verdadeira (V ou 1)
      - "N" Condição invalida (F ou 0)
      - "- "Condição não importa (N/A)

- Para ações:
    - “X” Ação deve ocorrer (S ou V ou 1)
    - Em branco a ação não deve ocorrer (- ou N ou F ou 0)

- Tabela completa são todas as condições
    - Resposta elevado as condições (Exemplo ao lado = 2^3)

Tabela de Decisão

Para que seja definida a quantidade de regras da tabela, basta que multipliquemos a quantidade de respostas possíveis de cada condição.

Exemplo:

- Condição 1 sim/não: = 2
- Condição 2 sim/não: = 2
- Condição 3 Sim/não: = 2

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

– Quantidade de Regras: = 2 x 2 x 2 = 8

- Tabela reduzida testar cada condição negativamente uma vez + o restante teste positivo (Número de ações (Sim ou Não) + 1)
    - Exemplo: Regra 2, Regra 3 e Regra 5

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- Exemplo:
  - área de condições;
  - área de ações;
  - Regras de decisão.

Foi testada negativamente uma vez e o restante positivo

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.

- Cobertura mínima é ter pelo menos um caso de teste por regra na decisão da tabela
- Cobertura é medida como o número de regras de decisão testadas por pelo menos um caso de teste, divido pelo número total de regras de decisão
- Pode ser aplicada a todas as situações em que o comportamento do software depende de uma combinação de condição, em qualquer nível de teste.

- Teste transição de estado
  - Componentes ou sistemas podem responder de maneira diferente a um evento dependendo das condições atuais ou do histórico anterior, fica conhecido como conceitos de estado.
  - Diagrama de transição de estado mostra os possíveis estados do software, bem como a forma como o software entra e sai e transita entre os estados.
  - Uma transição é iniciada por um evento
  - O evento resulta em transição
  - Se o mesmo evento pode resultar em duas ou mais transições diferentes do mesmo estado, esse evento pode ser qualificado por uma condição de proteção
  - Mudança de estado pode fazer com que o software executa uma ação
  - Uma tabela de transição de estado mostra todas as transições validas e potencialmente inválidas entre estados, bem como os eventos, condições de proteção e as ações resultantes para transição validas.
  - Diagrama de transição de estado normalmente mostram apenas as transições validas e excluem as transições invalidas

- Os testes podem ser projetados para cobrir uma series de situações especificas
- Os testes são baseados em linha de negócios associados a transições
- Amplamente usado na indústria software de prateleira
- O conceito de estado é abstrato – pode representar algumas linhas de código ou processo de negócios interno
- A cobertura é medida como o número de estados identificados ou transições testadas, dividido pelo número total de estados ou transições identificadas no objeto do teste.
- Imagem de ajuda
    - Transição de estado

- **Estado:** são as condições ou situações durante a vida de um sistema na qual ele satisfaz algumas condições, executa algumas atividades ou espera por eventos.
- **Transição:** é o relacionamento entre dois estados, indicando que o objeto que está no primeiro estado irá passar para o segundo estado mediante a ocorrência de um determinado evento.
- **Evento:** é causa necessária para que haja a transição de estado.
- **Ação**: A ação é basicamente a resposta que o sistema dará ao evento executado.
- **Estado inicial**: estado por onde se começa a leitura de um diagrama de estado.
- **Estado final**: estado que representa o fim.

- Teste caso de uso
    - Os testes podem ser derivados de casos de uso, que são uma maneira especifica de projetar interações com itens de software, incorporando requisitos para as funções de software representadas pelos casos de uso.
    - Casos de uso estão associados a atores (Usuário humanos ou componentes do sistema ou sistemas) e assuntos (Componente ou sistemas ao qual o caso de uso é aplicado)
    - Cada caso de uso especifica algum comportamento que um assunto pode realizar em colaboração com um ou mais atores.

- Caso de uso pode ser descrito por interações e atividades, bem como condições prevista, pós condições e linguagem natural.
- Interações entre os atores e o sujeito podem resultar em mudança no estado do sujeito.
- Podem ser representadas graficamente por fluxos de trabalho, diagramas ou modelos de processos.
- Pode incluir possíveis variações de comportamento básico, incluindo excepcional e tratamento de erros
- Os testes são projetados para exercitar os comportamentos definidos
- A cobertura pode ser medida de pela porcentagem de comportamento de casos de uso testados dividida pelo número total de comportamento

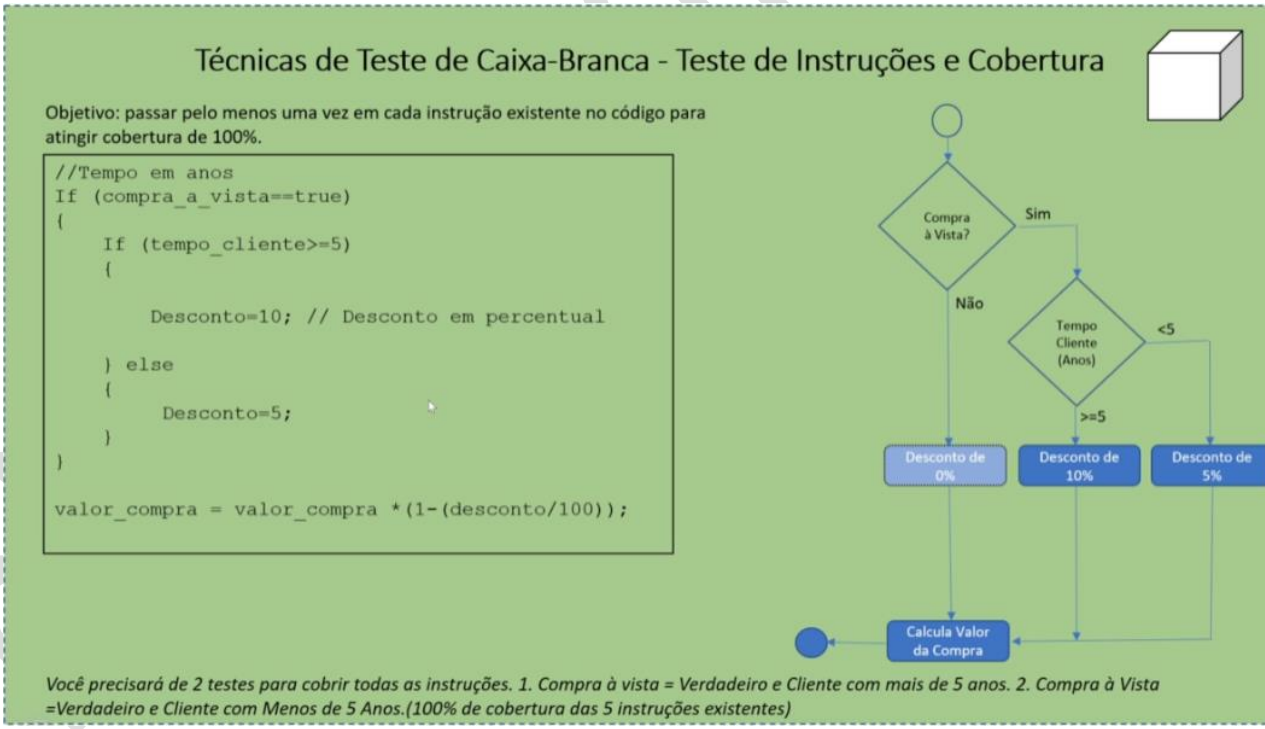
Teste Caixa Branca

- Baseado na estrutura interna do objeto de teste
- Pode ser usado em todos os níveis de teste
- Existem algumas técnicas mais avançada que são usadas em ambientes de segurança critica, de missão crítica ou de alta integridade.
- Passam muito pelo conceito de cobertura (Todas as técnicas de caixa branca começam com o nome “Cobertura”)
- Teste e cobertura de instrução (Cobertura de sentença, declaração e instrução)
  - Testa todas as instruções do código
  - Sempre é menor ou igual ao teste de cobertura de decisão
  - Quando tiver 2 ou mais conjuntos, o maior prevalece
- Teste e cobertura de decisão (Desvio)
  - Testa as decisões existentes no código e o código executado com base nos resultados da decisão
  - Os casos de teste seguem o fluxo de controle que ocorrem de um ponto de decisão.
  - Complexidade ciclo matica de McCabe (Cobertura de Decisão)
- O valor da instrução e teste de decisão
  - Quando a cobertura de 100% das instruções é alcançada, garante-se que todas as instruções executáveis no código tenham sido testadas, mas não garante que toda a lógica de decisão tenha sido testada.
  - Das duas técnicas caixa-branca discutida neste syllabus, o teste de instrução é o que fornece menos cobertura do que o teste de decisão

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- Quando uma cobertura de decisão atinge de 100% é alcançada, representa-se que todos os resultados de decisão foram testados, o que pode incluir resultados verdadeiros e falso.
- Cobertura de instrução ajuda a encontrar defeitos no código que não foi exercido por outros testes
- Cobertura de decisão ajuda a encontrar defeitos no código, em que outros testes não obtiveram resultados falsos e verdadeiros.
- Atingir 100% de cobertura de decisão garante 100% de cobertura de instrução (mas não vice versa)
- Cobertura de Instrução (Cobertura de sentença, declaração e comando)
- Cobertura de decisão (Cobertura de desvio)
- Complexidade ciclo matica de McCabe

- Resumo teste de cobertura e decisão
  - CI <= CD



Teste Baseado em experiencia

- Dependendo da abordagem do testador e experiencia, essas técnicas podem alcançar graus de cobertura e eficácia amplamente variados
- A cobertura pode ser difícil de avaliar e pode não ser mensurável.
- Suposição de erro:
  - É uma técnica usada para prever a ocorrência de erros, defeitos e falhas, com base no conhecimento:

        - Como o aplicativo funcionou no passado
        - Que tipos de erro tendem a ser cometidos
        - Falhas ocorridas em outros aplicativos
    - Uma abordagem metódica para a técnica de suposição de erro é criar uma lista de possíveis erros, defeitos e falhas e modelar os testes que exporão falhas e os defeitos que as causaram

- Teste Exploratório (Cartas de testes)
    - Testador experiente, planejamento leve que produza carta de teste e fazer o teste em um limite de tempo
    - Muito comum no desenvolvimento ágil
    - Os testes informais são modelados, executados, registrados e avaliados dinamicamente durante a execução do teste
    - Os resultados são usados para aprender mais sobre o componente ou sistema e para criar teste para as áreas que podem precisar de mais teste
    - É realizado usando testes baseados em sessão para estruturar as atividades de teste
    - Teste baseado em sessão, o teste exploratório é conduzido dentro de uma janela de tempo definida, e o testador usa um termo contendo objetivos de teste para orientar o teste.
    - É mais útil quando há poucas ou inadequadas especificações ou pressão de tempo significativo nos testes
    - Também é útil para complementar outras técnicas de teste mais formais
    - Pode ser usado em outras técnicas de teste de caixa-preta, caixa-branca e experiencia.
- Teste baseado em checklist:
    - Os testadores modelam, implementam e executam testes para cobrir as condições de teste encontradas em uma lista.
    - Testadores criam uma lista ou expandem uma existente, mas os testadores também podem usar um checklist existente sem modifica-lo
    - Checklist podem ser construídos com base na experiencia, conhecimento sobre o que é importante para o usuário ou uma compreensão de por que e como o software falha
    - Checklist podem dar suporte a vários tipos de teste, incluindo funcionais e não funcionais
    - Na ausência de teste detalhado, os checklist podem fornecer detalhes e consistência.

Capitulo 05 Gerenciamento de teste

Teste independente

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- As tarefas de teste podem ser executadas por pessoas em um papel especifico ou por pessoas em outros papeis
- Um certo grau de independência geralmente trona o testador mais eficaz em encontrar os defeitos devido as diferenças entre os vieses cognitivos do autor e do testador
- A independência não é, no entanto, um substituto para a familiaridade com o produto, e os desenvolvedores podem encontrar com eficiência muitos defeitos em seu próprio código.
- Graus de independência no teste incluem (5 no Total)
    - Sem testadores independentes, a única forma de teste disponível são os desenvolvedores testando 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.
    - Equipe de teste independente ou grupo dentro da organização, reportando-se ao gerenciamento de projetos ou ao gerenciamento executivo
    - Testadores independentes da organização empresarial ou da comunidade de usuários, ou com especialização em tipos de testes específicos, como usabilidade, segurança, performace, regulamentação, conformidade ou portabilidade
    - Testadores independentes externos a organização, trabalhando dentro ou fora do escritório
- Para maioria dos tipos de projetos, geralmente é melhor ter vários níveis de teste, com alguns desses níveis gerenciados por testadores independentes.
- Os desenvolvedores devem participar dos testes especialmente nos níveis mais baixos.
- Maneira pela qual a independência dos testes é implementada varia de acordo com o ciclo de vida do desenvolvimento do software.
- Benefícios potenciais da independência:
    - Os testadores independentes provavelmente reconhecerão diferentes tipos de falhas em comparação a os desenvolvedores, devido a diferentes históricos, perspectivas técnicas e vieses.
    - Um testador independente pode verificar, desafiar ou refutar as suposições feitas pelos stakeholders durante a especificação e a implementação do sistema.
- As desvantagens potenciais da independência:
    - Isolamento da equipe de desenvolvimento
    - 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 informações importantes
- Tarefas de um gerente de teste e do testador

- Gerente de teste:
    - Processo de teste e da liderança das atividades bem-sucedidas de teste
    - Pode ser executado por um gerente de teste profissional ou por um gerente de projeto, um gerente de desenvolvimento ou um gerente de qualidade.
    - Em projetos ou organizações maiores, várias equipes de teste podem se reportar a um gerente de teste, um técnico de teste ou u coordenador de teste, cada equipe sendo liderada por um líder de teste ou um testador líder.
    - Tarefas típicas de gerente de teste:
        - Desenvolver ou revisar uma política de teste e uma estratégia de teste para a organização
        - Planejar as atividades de teste
        - Escrever atualizar os planos de teste
        - Coordenar os planos de teste com gerente de projetos
        - Compartilhar as perspectivas de teste com outras atividades do projeto
        - Iniciar a analise, o projeto, a implementação e a execução dos testes, monitorar o progresso e os resultados obtidos e verificar o status dos critérios de saída
        - Preparar e entregar os relatórios de progresso do teste e resumo com base nas informações coletadas
        - Adaptar o planejamento com base nos resultados e no progresso dos testes e tomar as ações necessárias par ao controle de teste.
        - Suporte para configurar o sistema de gerenciamento de defeitos e o gerenciamento de configurações adequado do testware.
        - Introduzir as métricas adequadas para medir o progresso do teste e avaliar a qualidade do teste e do produto
        - Apoia a seleção e implementação de ferramentas para apoiar o processo de teste, incluindo a recomendação orçamentaria para a seleção de ferramentas.
        - Decidir sobre a implementação dos ambientes de teste
        - Promover e defender os testadores, a equipe de teste e a profissão de teste dentro da organização
        - Desenvolver habilidades e carreiras dos testadores.
        - A maneira como o gerente de teste desempenha sua função pode varias do modelo de ciclo de vida do software
- Tarefas típicas de testador:
    - Revisar e contribuir para os planos de teste

- Analisar, revisar e avaliar os requisitos, as histórias de usuários e os critérios de aceite, as especificações e modelos para estabilidades.
- Identificar e documentar as condições de teste e capturar a rastreabilidade entres os casos de teste, as condições de teste e a base de tese
- Projetar, configurar e verificar os ambientes de teste, geralmente coordenando com a administração do sistema e gerenciamento da rede
- Projetar e implementar os casos de teste e procedimento de teste
- Preparar e adquirir os dados de teste
- Criar o cronograma detalhado de execução do teste
- Executar os testes, avaliar os resultados e documentar os desvios de resultados esperados.
- Usar ferramentas apropriadas para facilitar o processo de teste
- Automatizar os testes conforme necessários
- Avaliar as características não funcionais
- Revisar os testes desenvolvidos por outros
- Pessoas diferentes podem assumir o papel de testador em diferentes níveis de teste.

Planejamento e estimativa de testes

- Planejamento é influenciado pela política de teste e a estratégia de teste da organização, pelos ciclos de vida e métodos, pelo escopo dos testes, objetivos, riscos, restrições, criticidade, estabilidade e disponibilidade de recursos
- O planejamento do teste é uma atividade continua e é executado durante todo o ciclo de vida do produto.
- Planejamento pode ser documentado em um plano de teste principal e em planos de teste separados para cada nível de teste
- As atividades do planejamento do este podem ser documentados em um plano de teste e incluir:
  - Determinar o escopo, os objetivos e os riscos do teste
  - Definir a abordagem do teste
  - Integrar e coordenar as atividades de teste nas atividades do ciclo de vida do software
  - Tomar decisões sobre o que testar, as pessoas e outros recursos necessários para realizar as várias atividades de teste e como essas atividades serão realizadas
  - Programar as atividades de analise, projeto, implementação, execução e avaliação de teste, em datas especificas
  - Selecionar as métricas para monitoramento e controle de teste

        - Orçar as atividades de teste
        - Determinar o nível de detalhes e a estrutura da documentação de teste
    - Tipos comuns de estratégias de teste (ABC2DMR)
        - Analítica
            - Teste é baseado em uma análise de algum fator
        - Baseada em modelo
            - Teste são projetados com base em algum modelo de algum aspecto necessário do produto, como uma função, um processo empresarial, uma estrutura interna ou uma característica não funcional
        - Metódica
            - Depende do uso sistemático de um conjunto predefinido de testes ou condições de teste
        - Compatível com processo
            - Envolve analise, projeto e implementação do teste baseado em regras e padrões externos
        - Dirigida
            - Orientado principalmente pelo aconselhamento, orientação ou instrução dos stakeholders, especialista no domínio do negócio ou especialista em tecnologia.
        - Contra regressão
            - Motivado pelo desejo de evitar a regressão de recursos existentes. Essa estratégia inclui a reutilização do testware existente, da automação extensiva de regressão e de conjunto de teste padrão
        - Reativa
            - Teste é reativo ao componente ou sistemas que está sendo testado e aos eventos que ocorrem durante a execução do teste, em vez de serem pré-planejados. Teste exploratório é uma técnica comum empregada em estratégias salvas.
    - Uma estratégia de teste apropriada é frequentemente criada pela combinação de vários desses tipos de estratégias de teste.
    - Abordagem de teste é o ponto de partida para selecionar as técnicas de teste, os níveis de teste e os tipos de teste, e para definir os critérios de entrada e saída.
    - A adaptação da estratégia baseia-se em decisões tomadas em relação a complexidade e objetivos do projeto, do tipo de produto que está sendo desenvolvido e da análise de risco do produto.
    - A abordagem selecionada depende do contexto e pode considerar fatores como riscos, seguranças, recursos e habilidades disponíveis, tecnologia, natureza do sistema.

- Critérios de entradas e saídas:
    - Para exercer o controle efetivo sobre a qualidade do software e dos testes, é aconselhável ter critérios que definam quando uma determinada atividade de teste deve iniciar e quando a atividade é concluída.
    - Os critérios de entrada definem as condições previas para a realização de uma determinada atividade de teste
    - Critérios de saída definem quais condições devem ser atingidas para declarar que um nível de teste ou um conjunto de teste foram concluídos.
    - Critérios de entrada e saída devem ser definidos para cada nível e tipo de teste e serão diferentes com base nos objetivos do teste
    - Critérios de entrada:
        - Disponibilidade dos requisitos testáveis, histórias de usuários e/ou modelos
        - Disponibilidade dos itens de teste que atendam aos critérios de saída para quaisquer níveis de teste anteriores
        - Disponibilidade do ambiente de teste
        - Disponibilidade de ferramentas de teste necessárias
        - Disponibilidade de dados de teste e outros recursos necessários.
    - Critérios de saída:
        - Que os teste planejados forame executados
        - Foi alcançado um nível definido de cobertura
        - O número de defeitos não resolvidos está dentro de um limite acordado
        - O número de defeitos remanescente estimados é suficientemente baixo
        - Os níveis avaliados de confiabilidade, eficiência de performace, usabilidade, segurança e outras características de qualidade relevantes são suficientes
    - Mesmo sem que os critérios de saídas sejam satisfeitos, também é comum que as atividades de teste sejam reduzidas devido ao orçamento gasto, a conclusão do prazo programado ou da pressão para levar o produto ao mercado
    - Pode ser aceitável encerrar os testes sob tais circunstancias, se os stakeholders e os donos das empresas assumirem o risco de entrar em vigor sem mais teste
- Cronograma de execução do teste
    - Depois que vários casos de teste e procedimentos de teste são produzidos e montados em suítes de teste, os conjuntos de teste podem ser organizados em um cronograma da execução do teste que define a ordem em que devem ser executados
    - O cronograma deve levar em consideração fatores como priorização, dependência, teste de confirmação, teste de regressão e a sequência mais eficiente para se executar os testes.

        - O ideal seria que os casos de teste fossem ordenados para serem executados com base em seus níveis de prioridade, geralmente executando os casos de teste com a prioridade mais alta primeiro
        - A pratica acima pode não funcionar se os casos de teste tiverem dependência ou os recursos que estão sem testado tiverem dependências.
        - Os testes de confirmação e regressão também devem ser priorizados, com base na importância de feedback rápido sobre as mudanças, mas vim aqui novamente as dependências podem ser aplicadas
    - Fatores que influenciam o esforço do teste
        - A estimativa do esforço do teste envolve a previsão da quantidade de trabalho relacionado ao teste que será necessário para atender aos objetivos do teste de um projeto, release ou iteração em particular
        - Os fatores que influenciam o esforço do teste podem incluir características:
            - Característica do produto:
                - Riscos associados com o produto
                - Qualidade da base de teste
                - Tamanho do produto
                - Complexidade do domínio do produto
                - Requisitos das características de qualidade
                - Nível necessário de detalhes para documentação de teste
                - Requisitos para conformidade legal e regulatória
            - Característica do processo de desenvolvimento:
                - Estabilidade e maturidade da organização
                - O modelo de desenvolvimento em uso
                - A abordagem de teste
                - A ferramenta usada
                - O processo de teste
                - A pressão sobre o tempo de finalização
            - Característica e pessoas:
                - As habilidades e a experiência das pessoas envolvidas, especialmente com projetos e produtos semelhantes
                - Coesão e liderança de equipe
            - Resultados do teste:

        - O número e a gravidade dos efeitos encontrados
        - A quantidade de retrabalho necessário.
- Técnicas de estimativa de teste:
    - Técnica baseada em métricas:
        - Com base nas métricas de projetos anteriores, ou com base em valores típicos
    - Técnica baseada em especialista:
        - Com base na experiencias dois responsáveis pelas tarefas de teste ou por especialistas.
    - No desenvolvimento ágil usa-se o gráfico burndown, técnica baseada em métricas
    - Planning poker, técnica baseada em especialistas
    - Em projetos sequenciais os modelos de remoção de defeitos são exemplos de técnica baseadas em métricas.

Monitoramento e controle de teste

- Monitoramento (Acompanhar) X Controle (Agir)
- O objetivo é coletar informações e fornecer feedbacks e visibilidades sobre as atividades de teste
- As informações podem ser coletadas manualmente ou automaticamente
- O controle de teste descreve quaisquer ações orientadas ou corretivas tomadas como resultado de informações e métricas coletadas e possivelmente relatadas.
- As ações podem cobrir quaisquer atividades de teste e podem afetar qualquer outra atividade do ciclo de vida do software
- Exemplo ações de controle do teste:
    - Priorizar os testes quando ocorrer um risco identificado
    - Alterar o cronograma do teste devido a disponibilidade ou indisponibilidades de um ambiente de teste ou outros recursos
    - Reavaliar se um item de teste atende a um critério de entrada ou saída devido ao retrabalho
- Métricas usadas no teste
    - As métricas podem ser coletadas durante o final das atividades de teste para avaliar:
        - Relação entre o planejado e o orçado em cronograma
        - Qualidade atual do objeto de teste
        - Adequação da abordagem de teste

        - Eficácia das atividades de teste em relação aos objetivos.
    - As métricas de teste comum incluem:
        - Porcentagem de trabalho planejado executado na preparação do caso de teste
        - Porcentagem do trabalho planejado executado na preparação do ambiente de teste
        - Execução de caso de teste
        - Informação sobre defeitos
        - Cobertura de teste de requisitos, de histórias de usuários, de critérios de aceite, de risco ou de código
        - Conclusão de tarefas, alocação e uso de recursos e esforços
        - Custo do teste, incluindo o custo comparada ao benefício de encontrar o próximo defeitos ou o custo comparado ao benefício de executar o próximo teste.
- Finalidades, conteúdo e público-alvo para os relatórios de teste
    - O objetivo do relatório de teste é resumir e comunicar informações de atividades de teste, durante e no final de uma atividade de teste
    - O relatório de teste, pode ser referido como um relatório de progresso do teste, enquanto um relatório de teste preparado no final de uma atividade de teste pode ser referido como um relatório de resumo do teste.
    - Relatórios de progresso de teste típicos podem incluir:
        - O status das atividades de teste e o progresso em relação ao plano de teste
        - Fatores impedindo o avanço
        - O teste planejado para o próximo período do relatório
        - A qualidade do objeto de teste
    - Quando os critérios de saída são atingidos, o gerente de teste faz o relatório de resumo do teste
    - Este relatório fornece um resumo dos testes realizados, com base no último relatório de progresso de teste e qualquer outra informação relevante.
    - Relatórios típicos de progresso de teste e relatórios:
        - Resumo dos testes realizados
        - Informação sobre o que ocorreu durante um período de teste
        - Desvios do plano, incluindo desvios no cronograma, duração ou esforço das atividades de teste

- Status do teste e da qualidade do produto com relação aos critérios de saída ou definição de completo
- Fatores que bloquearam ou continuam bloqueando o progresso
- Métricas de defeitos, caso de teste, cobertura de teste, progresso da atividade e consumo de recursos
- Riscos residuais
- Produtos de teste reutilizáveis produzido

    - O conteúdo de um relatório de teste irá várias dependendo do projeto, dos requisitos organizacionais e do ciclo de vida de desenvolvimento de software
    - Personalizar os relatórios de teste com base no contexto do projeto, os relatórios de teste devem ser personalizados com base no público do relatório
    - O tipo e a quantidade de informações que devem ser incluídas para uma audiência técnica ou uma equipe de teste podem ser diferentes daquelas que seriam incluídas em um relatório de resumo executivo

Gerenciamento de configurações

- Tem como objetivo é estabelecer e manter a integridade do componente ou do sistema, o testware e seus relacionamentos entre si durante o ciclo de vida do projeto e do produto
- Gerenciamento de configurações pode envolver o seguinte:
    - Todos os itens de teste são identificados de forma exclusiva, controlados por versão, rastreados para alterações e relacionados entre si
    - Todos os itens de testware são identificados de forma exclusiva, controlados por versão, rastreados para alterações, relacionados uns aos outros e relacionados as versões dos itens de teste, de forma que a rastreabilidades possa ser mantida durante todo o processo de teste
    - Todos os documentos e itens de software identificados são referenciados sem ambiguidade na documentação de teste

Riscos e testes

- Definição de risco
    - R = P x I (Risco = Probabilidade X Impacto)
    - O risco envolve a possibilidade da ocorrência futura de um evento com consequência negativas
    - O nível de risco é determinado pela probabilidade do evento e pelo impacto desse evento.
- Riscos de produto e projeto
    - O risco do produto envolve a possibilidade de que um produto de trabalho possa falhar em satisfazer as necessidades legitimas de seus usuários ou stakeholders

- Quando os riscos do produto estão associados as características especificas de qualidade do produto também são chamados de riscos de qualidade
- Riscos do produto incluem:
    - O software pode não executar as funções de acordo com a sua especificação
    - O software pode não executar as funções pretendidas de acordo com as necessidades do usuário, do cliente ou dos stakeholders.
    - Uma arquitetura de sistema pode não suportar adequadamente alguns requisitos não funcionais
    - Um cálculo especifico pode ser executado incorretamente em alguma circunstância.
    - Uma estrutura de controle de loop pode ser codificada incorretamente
    - Os tempos de reposta podem ser inadequados para um sistema de processamento de transações de alta performace
    - O feedback da experiencia do usuário pode não atender as expectativas do produto
- O risco do projeto envolve situações que, caso ocorram, podem ter um efeito negativo na capacidade de um projeto atingir seus objetivos
- Questões do projeto:
    - Atrasos podem ocorrer na entrega, na conclusão da tarefa ou na satisfação dos critérios de saída ou definição de feito
    - Estimativas imprecisas, realocação de fundos para projetos de maior prioridade ou corte geral de custos na organização podem resultar em recursos financeiros inadequados
    - Alterações tardias podem resultar em um substancial retrabalho
- Questões organizacionais:
    - Insuficiência de equipe, habilidades e treinamentos
    - Questões pessoais podem causar problemas e conflitos
    - Prioridades comerciais conflitantes podem causar indisponibilidade de usuário, equipe de negócios ou especialistas no assunto
- Questões políticas:
    - Os testadores podem não comunicar adequadamente suas necessidades ou os resultados do teste
    - Os desenvolvedores ou testadores podem falhar em acompanhar as informações encontradas nos testes e revisões
    - Pode haver uma atitude impropria em relação as expectativas de teste

- Questões técnicas:
    - Os requisitos podem não estar bem definidos
    - Os requisitos podem não ser cumpridos por restrições existentes
    - O ambiente de teste pode não estar pronto no prazo
    - A conversão de dados, o planejamento de migração e o suporte de ferramentas podem atrasar
    - Fraquezas no processo de desenvolvimento podem afetar a consistência ou a qualidade dos produtos de trabalho do projeto, como desenho, código, configuração, dados de teste e casos de teste
    - Má gestão de defeitos e problemas semelhantes podem resultar em defeitos acumulados e outras obrigações técnicas.
- Questões de fornecedores:
    - Um terceiro pode deixar de entregar um produto ou serviço necessário ou ir à falência
    - Questões contratuais podem causar problemas no projeto
- Os riscos do projeto podem afetar tanto as atividades de desenvolvimento quanto as atividades de teste.
- Os gerentes de projeto são responsáveis por lidar com todos os riscos do projeto, mas não é incomum que os gerentes de teste tenham responsabilidades pelos riscos do projeto relacionados ao teste.

- Teste baseado em risco e qualidade do produto
    - O risco é usado para concentrar o esforço necessário durante o teste
    - Ele é usado para decidir onde e quando começar a testar e identificar áreas que precisam de mais atenção
    - O teste é usado para reduzir a probabilidade da ocorrência de um evento adverso ou para reduzir seu impacto.
    - O teste é usado como uma atividade de mitigação de risco, para fornecer feedback sobre os riscos identificados, bem como sobre os riscos residuais.
    - Uma abordagem e teste baseada em risco fornece oportunidades proativas para reduzir os níveis de risco do produto
    - Envolve a análise de risco, que inclui a identificação e a avaliação da probabilidade e impacto de cada risco.
    - As informações resultantes sobre o risco do produto são usadas para orientar o planejamento, a especificação, a preparação e a execução dos casos de teste, além do monitoramento e controle teste
    - Em uma abordagem de teste baseada em risco, os resultados da análise de risco do produto são usados para:

        - Determinar as técnicas de teste a serem empregadas
        - Determinar os níveis e tipos específicos de teste a serem realizados.
        - Determinar a extensão do teste a ser realizado
        - Priorizar o teste na tentativa de encontrar antecipadamente os defeitos críticos
        - Determinar se quaisquer atividades além do teste poderiam ser empregadas para reduzir o risco.
    - O teste baseado em risco baseia-se no conhecimento coletivo e no conhecimento de stakeholders do projeto para realizar a análise de risco do produto.
    - As atividades de gerenciamento de risco fornecem uma abordagem disciplinadas para:
        - Analisar (e reavaliar regulamente) o que pode dar errado
        - Determinar quais os riscos são importantes para lidar
        - Implementar ações para atenuar esses riscos
        - Planejar a contingência para lidar com os riscos, caso eles se tornem eventos reais
    - Os testes podem identificar novos riscos, ajudar a determinar quais os riscos devem ser atenuados e reduzir a incerteza sobre os riscos.

Gerenciamento de defeitos

- A maneira como os defeitos são registrados pode variar, dependendo do contexto do componente ou sistema que está sendo testado, do nível de teste e do modelo de ciclo de vida de desenvolvimento de software
- Quaisquer defeitos identificados devem ser investigados e rastreados desde a descoberta e classificação até sua resolução
- Para gerenciar todos os defeitos, uma organização deve estabelecer um processo de gerenciamento de defeitos que inclua um fluxo de trabalho e regras sua classificação.
- Esse processo deve ser acordado com todos os participantes do gerenciamento de defeitos
- Em algumas organizações, o registro e o rastreamento de defeitos podem ser muito informais
- Durante o processo de gerenciamento de defeitos, alguns dos relatórios podem revelar falsos positivos, e não falhar reais devido aos defeitos
- Os testadores devem minimizar o número de falsos positivos relatados como defeitos
- Os defeitos podem ser relatados durante a codificação, análise estática, revisões, testes dinâmicos ou uso de um produto de software.
- Os defeitos podem ser relatados em qualquer tipo de documentação
- Para ter um processo eficaz e eficiente de gerenciamento de defeitos, as organizações podem definir padrões para os atributos, classificação e fluxo de trabalho de defeitos.

- Relatório de defeitos tem os seguintes objetivos:
    - Fornece aos desenvolvedores e a outros, informações sobre qualquer evento adverso ocorrido, para que possam identificar efeitos específicos, isolar o problema com um teste mínimo de reprodução e corrigir o defeito potencial, conforme necessário ou resolver o problema
    - Fornecer aos gerentes de teste um meio de rastrear a qualidade do produto de trabalho e o impacto do teste
    - Fornecer ideias para desenvolvimento e melhoria de processo de teste
- Relatório de defeitos arquivados durante o teste dinâmico inclui:
    - Um identificador
    - Um título e um breve resumo do defeito relatado
    - A data do relatório de defeitos, a organização emissora e autor
    - A identificação do item de teste e ambiente
    - As fases do ciclo de vida de desenvolvimento em que o defeito foi observado
    - Uma descrição do defeito para permitir a reprodução e resolução, incluindo logs, capturas de tela dum de banco de dados ou gravações
    - Resultados esperados e reais
    - Escopo ou grau de impacto do defeito sobre os interesses dos stakeholders
    - Urgência/prioridade para corrigir
    - O estado do relatório de defeitos
    - Conclusões, recomendação e aprovações
    - Questões globais, como outras áreas que podem ser afetadas por uma alteração resultante do defeito
    - Histórico de alterações, como a sequência de ações tomadas pelos membros da equipe do projeto com relação ao defeito para isolar, reparar e confirmar como corrigido
    - Referencias, incluindo o caso de teste que revelou o problema
- Alguns desses detalhes podem ser automaticamente incluídos ou gerenciados ao usar ferramentas de gerenciamento de defeitos.
- Os defeitos encontrados durante os testes estáticos, particularmente revisões, normalmente serão documentados de uma maneira diferente.

Capitulo 06 Ferramenta de suporte ao teste

Considerações sobre ferramentas de teste

- As ferramentas de teste podem ser usadas para suportar uma ou mais atividades de teste. Tais ferramentas incluem:

- Ferramentas são usadas diretamente nos testes, como ferramentas de execução e ferramentas de preparação de dados de teste
- Ferramentas que ajudam a gerenciar os requisitos, os casos de teste, os procedimentos de teste, os scripts de teste automatizados, os resultados de testes, os dados de teste e defeitos, e para os relatórios e monitoramento da execução do teste
- Ferramentas são usados para investigação e avaliação
- Qualquer ferramenta que auxilie no teste.
- Classificação das ferramentas de teste
    - As fermentas de teste podem te rum ou mais dos seguintes propósitos, dependendo do contexto:
        - Melhorar a eficiência das atividades de teste automatizando tarefas repetitivas ou tarefas que exigem recursos significativos quando feitas manualmente.
        - Melhorar a eficiência das atividades de teste, apoiando as atividades de teste manuais durante todo o processo.
        - Melhorar a qualidade das atividades de teste, permitindo testes mais consistentes e um nível mais alto de reprodutibilidade de defeitos
        - Automatizar as atividades que podem ser executadas manualmente
        - Aumentar a confiabilidade dos testes
    - As fermentas podem ser classificadas com base em vários critérios, como finalidade, preço, modelo de licenciamento e a tecnologia utilizada.
    - Algumas ferramentas suportam claramente apenas ou principalmente uma atividade, outros podem apoiar mais de uma atividade.
    - As ferramentas de um único provedor, podem ser fornecidas como um conjunto integrado
    - Alguns tipos de ferramentas de teste podem ser intrusivos, o que significa que podem afetar o resultado real do teste
    - Ramo de Tópico
    - Algumas ferramentas ofertem suporte que é tipicamente mais apropriado para desenvolvedores
    - Essas ferramentas estão marcadas com “[D]” nas seções.
    - Ferramentas de suporte para gerenciamento de teste e testware
        - Ferramentas de gerenciamento podem ser aplicadas em qualquer atividade de teste durante todo o ciclo de vida de desenvolvimento de software:
            - Ferramentas de gerenciamento de teste e ferramentas de gerenciamento do ciclo de vida de aplicativo
            - Ferramentas de gerenciamento de requisitos

        - Ferramentas de gerenciamento de defeitos
        - Ferramentas de gerenciamento de configuração
        - Ferramentas de integração continua [D]
    - Ferramentas de suporte ao teste estático
        - Ferramentas de teste estático estão associadas as atividades e benefícios descritos no capitulo 3
            - Ferramentas de análise estática [D]
    - Ferramentas de suporte para modelagem e implementação dos testes
        - Ferramentas de projetos de teste auxiliam na criação de produtos de trabalho sustentáveis no projeto e implementação dos testes:
            - Ferramentas de teste baseadas em modelo
            - Ferramentas de preparação de dados de teste
        - Em alguns casos, as ferramentas que suportam a modelagem e a implementação dos testes também podem suportar a execução e o registro de teste
    - Ferramentas de suporte para execução e registro de teste
        - Existem ferramentas para apoiar e aprimorar as atividades de execução e registro de teste:
            - Ferramentas de execução do teste
            - Ferramentas de cobertura
            - Ferramentas de teste [D]
    - Ferramentas de suporte para medição de performace e análise dinâmica
        - As ferramentas de medição de perfomace e analise dinâmica são essenciais para dar suporte as atividades de perfomace e de teste de carga, pois essas atividades não podem ser feitas manualmente com eficiência:
            - Ferramentas de teste de performace
            - Ferramentas de análise dinâmica [D]
- Benefícios e riscos da automação de teste
    - Simplesmente adquirir uma ferramenta não garante o sucesso de seu projeto
    - Cada nova ferramenta introduzida em uma organização exigira esforço para obter benefícios reais e duradouros.
    - Existem benefícios e oportunidades potenciais com o uso de ferramentas nos testes, mas também há riscos

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com

- Isso é particularmente verdadeiro para as ferramentas de execução do teste (que geralmente são chamados de automação de teste)
- Benefícios potenciais:
    - Redução no trabalho manual repetitivo, economizando tempo
    - Maior consistência e repetibilidade
    - Avaliação mais objetiva
    - Acesso mais fácil a informação sobre teste
- Riscos potenciais:
    - Expectativas irreais para a ferramenta
    - O tempo, custo e o esforço para a implantação de uma ferramenta podem ser subestimados
    - O tempo e o esforço necessários pra obter os benefícios significativos e contínuos da ferramenta podem ser subestimados
    - O esforço necessário para manter os ativos de teste gerados pelas ferramentas pode ser subestimado
    - A ferramenta pode ser usada em demasia
    - O controle de versão dos ativos de teste pode ser negligenciado
    - Os relacionamentos e os problemas de interoperabilidade entre ferramentas críticas podem ser negligenciados, como ferramentas de gerenciamento de requisitos, ferramentas de gerenciamento de configuração, ferramentas de gerenciamento de defeitos e ferramentas de vários fornecedores.
    - O fornecedor da ferramenta pode sair do negócio, aposentando a ferramenta ou vendendo seu código para um fornecedor diferente
    - O fornecedor pode possuir um suporte insatisfatório a ferramenta, as atualizações e as correções de defeitos
    - Um projeto de código aberto pode ser suspenso
    - Uma nova plataforma ou tecnologia pode não ser suportada pela ferramenta
    - Pode não haver uma propriedade clara da ferramenta.

- Considerações especiais para execução de testes e ferramentas de gerenciamento de testes
    - Ferramentas de execução de teste executam objetos de teste usando scripts de teste automatizados
    - Esses tipos de ferramentas geralmente reque um esforço significativo para obter benefícios:
        - Abordagem captura de teste:

        - Capturar testes gravando as ações de um testador manual parece atraente, mas essa abordagem não pode ser dimensionada para muitos scripts de teste.
        - Um script capturado é uma representação linear com dados e ações específicos como parte de cada script.
        - Esse tipo de script pode ser instável quando ocorrem eventos inesperados e exigir manutenção continua à medida que a interface do usuário do sistema evolui com o tempo.
    - Abordagem de teste orientada por dados:
        - Essa abordagem separa as entradas de teste e os resultados esperados, geralmente em uma planilha, e usa um script de teste mais genéricos que pode ler os dados de entrada e executar o mesmo script de teste com dados diferentes
    - Abordagem de teste orientada por palavra-chave:
        - Essa abordagem de teste, um script genérico processa palavra-chave que descrevem as ações a serem executadas, que depois chama scripts de palavra-chave para processar os dados de teste associados.
- As abordagens acima exigem que alguém tenha experiencia na linguagem de script.
- Ao usar abordagens de testes controlados por dados ou por palavras-chaves, os testadores que não são familiarizados com a linguagem de script também podem contribuir criando dados de teste ou palavra-chave para esses scripts predefinidos.
- Independentemente da técnica de script usada, os resultados esperados para cada teste precisam ser comparados aos resultados reais do teste, dinamicamente (Enquanto o teste está em execução) ou armazenados para comparação posterior
- As ferramentas de teste baseado em modelo (MBT) permitem que uma especificação funcional seja capturada na forma de um modelo, como diagrama de atividades
- Essa tarefa geralmente é executada por um arquiteto de sistemas
- A ferramenta MBT interpreta o modelo para criar especificações de caso de teste que podem ser salvas em uma ferramenta de gerenciamento de teste ou executados por uma ferramenta de execução de teste
- As ferramentas de gerenciamento de teste geralmente precisam interagir com outras ferramentas ou planilhas por vários motivos:
    - Para produzir informações uteis em formato que atenda às necessidades da organização
    - Manter rastreabilidade consistente para os requisitos em uma ferramenta de gerenciamento de requisitos
    - Para vincular informação da versão do objeto de teste na ferramenta de gerenciamento de configuração

        - Isso é particularmente importante quando usar uma ferramenta integrada, que inclui um módulo de gerenciamento de teste, além de outros módulos usados por diferentes grupos dentro de uma organização

Uso eficaz de ferramentas

- As principais considerações para escolha de uma ferramenta ideal para uma organização incluem:
    - Avaliação da maturidade da organização, seus pontos fortes e fracos
    - Identificação de oportunidade para um processo de teste melhorado suportado por ferramentas
    - Compreensão das tecnologias usadas pelos objetos de teste, a fim de selecionas uma ferramenta que seja compatível com essa tecnologia
    - As ferramentas de integração continuam e construção já em uso dentro da organização, a fim de garantir a compatibilidade e integração de ferramentas
    - Avaliação da ferramenta contra requisitos claros e critérios objetivos.
    - Consideração sobre se a ferramenta está ou não disponível em um período de teste gratuito
    - Avaliação do fornecedor ou suporte para ferramentas não comerciais
    - Identificação de requisitos internos para treinamento e mentoring no uso das ferramentas.
    - Avaliação das necessidades de treinamento, considerando as habilidades de teste daqueles que trabalharão diretamente com as ferramentas.
    - Considerações de prós e contras de vários modelos de licenciamento
    - Estimativa de uma relação custo-benefício baseada em um caso de negócio concreto
    - Como etapa final, uma prova de conceito deve ser feita para estabelecer se a ferramenta funciona efetivamente com o software em teste e dentro da infraestrutura atual ou, se necessário, para identificar as alterações necessárias.
- Projeto piloto para introduzir uma ferramenta em uma organização
    - A introdução da ferramenta selecionada em uma organização geralmente começa com um projeto piloto, que tem os seguintes objetivos:
        - Obter conhecimento aprofundado sobre a ferramenta, entendo seus pontos fortes e fracos
        - Avaliar como a ferramenta se ajusta aos processos e praticas existentes e determinar o que precisar mudar
        - Decidir a forma padrão de usar, gerenciar, armazenar e manter a ferramenta e os recurso de teste
        - Avaliar se os benefícios serão alcançados a um custo razoável

            - Compreender as métricas que você deseja que a ferramenta colete e relate e configure a ferramenta para garantir que essas métricas possam ser capturadas e relatadas.

    - Fatores de sucesso para ferramentas
        - Fatores de sucesso para avaliação, implementação, implantação e suporte continuo de ferramentas dentro de uma organização incluem:
            - Estender o uso da ferramenta para o resto da organização de forma incremental
            - Adaptar e melhorar os processos para se adequar ao uso da ferramenta
            - Fornece treinamento, capacitação e orientação para os usuários das ferramentas
            - Definir as diretrizes para o uso da ferramenta
            - Implementar uma forma de coletar informações de uso real das ferramentas
            - Monitor o uso e os benéficos de ferramentas
            - Fornece suporte aos usuários de uma determinada ferramenta
            - Reunir lições aprendidas de todos os usuários
        - Também é importante garantir que a ferramenta seja técnica e organizacionalmente integrada ao ciclo de vida de desenvolvimento de software.

Criador: Thiago Andrade
Linkedin: https://www.linkedin.com/in/thiago-andrade-2a97555b/
Email: maciel.thiago@gmail.com