

Validação e Teste de Software

Quem sou?

Daniel Teófilo Vasconcelos

Data Analyst na Controladoria e Ouvidoria Geral do Estado do Ceará

Professor de Pós-Graduação na FBUi, Uni7 e Unifametro

Mestrado em Sistemas de Apoio à Decisão na UECE

18 anos na área de Tecnologia da Informação

O que é um Software?

O software é todo e qualquer aplicação que roda em uma máquina. No computador, os softwares são conhecidos como aplicações e são utilizados para diversas finalidades, por exemplo: a calculadora, o bloco de notas, o navegador, etc..

No celular, os softwares são chamados de aplicativos, os quais facilitam nosso dia a dia quando precisamos pedir comida, fazer uma ligação, trocar mensagens, assistir vídeos, entre outros.

Conceitos de Testes de Software

Teste de Software é um processo que faz parte do desenvolvimento de software, e tem como principal objetivo revelar falhas/bugs para que sejam corrigidas até que o produto final atinja a qualidade desejada / acordada.

Teste de Software é a investigação do software a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar, se relaciona com o conceito de verificação e validação.

“Um software é considerado confiável à medida que o número de defeitos diminui durante o seu uso.” - Myers^[1]

Teste de Software faz parte de um área mais abrangente chamada de Garantia da Qualidade de Software, cujo objetivo é verificar se todos os procedimentos relacionados à construção do software estão de acordo com os padrões de qualidade estabelecidos.

De acordo com Myres, testes continuam sendo a “**arte das trevas**” do desenvolvimento de software, ele percebeu que era comum profissionais de desenvolvimento entregarem softwares sem a realização de testes.

Quando pensamos em **documentação de sistema**, pode não ficar claro quais informações ela deve possuir. Geralmente não há um padrão formal de documentação. A documentação mais importante são as regras de negócio, pois elas são as regras do jogo.

Em uma documentação de sistema é essencial definir os possíveis cenários que o usuário poderá executar no software e as regras que restringem as operações dos usuários em cada tela.

A documentação formal pode consistir em **roteiros de teste**. Esses roteiros são compostos por um conjunto de casos de teste, que descrevem todos os passos necessários para executar alguma funcionalidade do sistema, e o resultado esperado após a execução dos passos.

A fim de reduzir o esforço na elaboração de roteiros de teste, é possível criar casos de teste genéricos. Por exemplo, independentemente do tipo de sistema, na autenticação de usuários pode-se criar testes para:

- Validar entrada com senha incorreta;
- Validar entradas de campos obrigatórios;
- Verificar autenticação com sucesso.

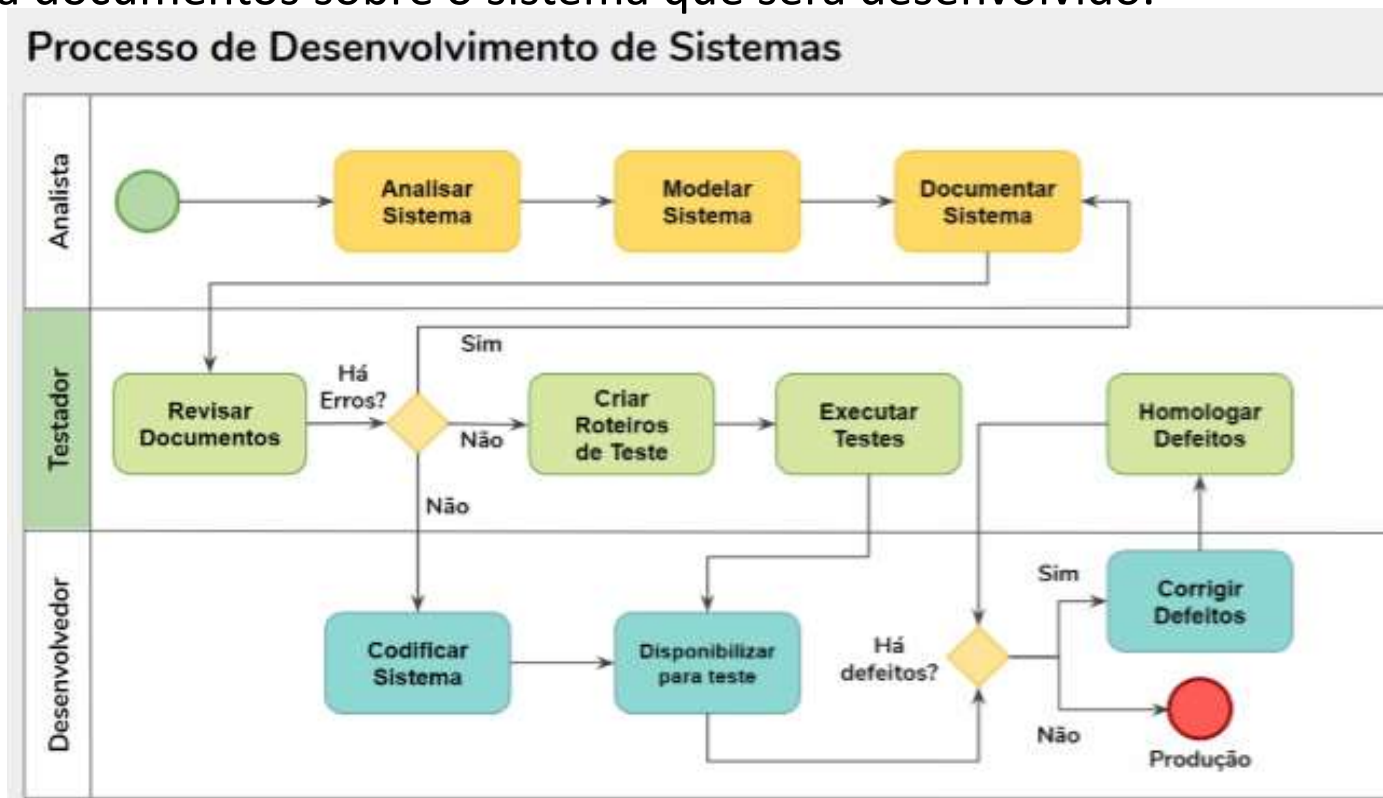
Em que momento os teste são realizados?

As atividades de testes de software são realizadas durante a construção e manutenção de um sistema. Para que essas atividades sejam realizadas de forma planejada é importante seguir um processo.

Um processo define um conjunto de atividades que podem ser executadas por cada membro de uma equipe durante o desenvolvimento de um produto.

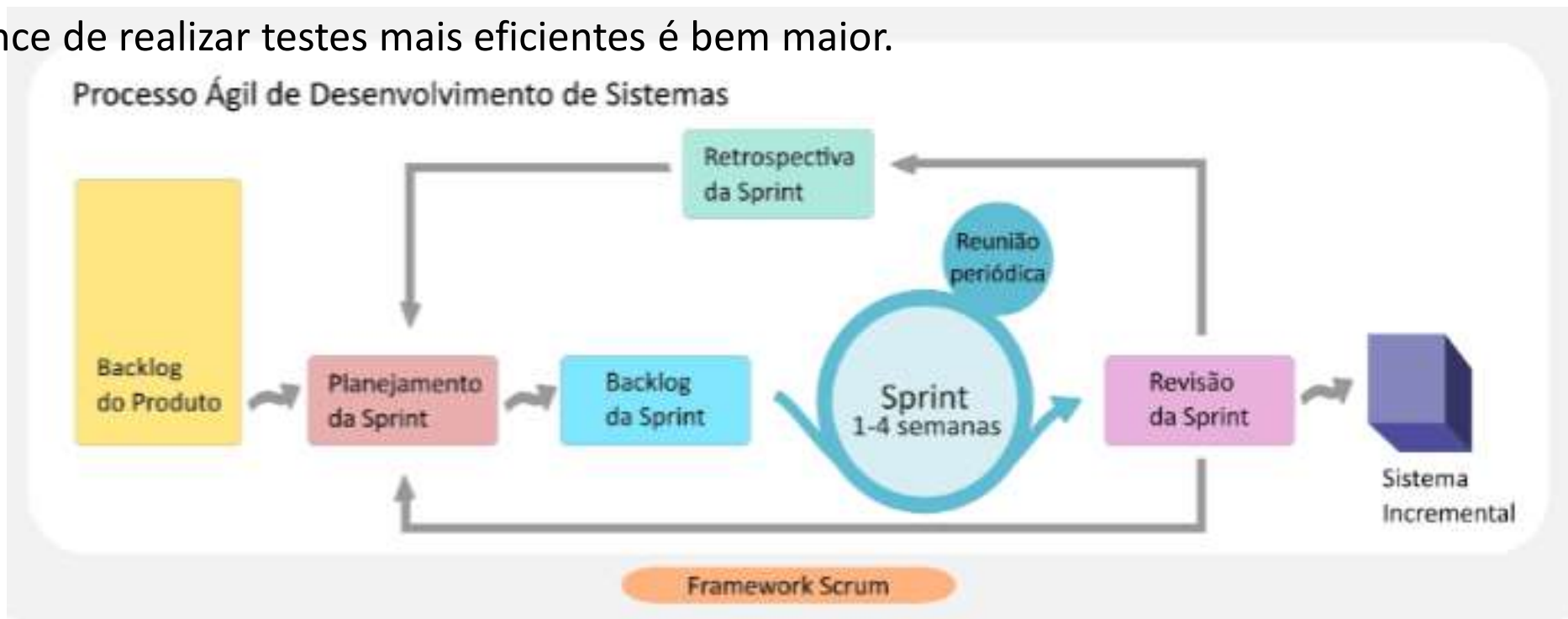
O processo de teste de software é parte do **processo de desenvolvimento** de sistemas, o qual pode ser um processo **tradicional** ou **ágil**.

O **processo de desenvolvimento tradicional** contempla de forma detalhada cada uma das atividades. Com isso, os artefatos produzidos por esse tipo de processo são mais completos e requerem mais conformidade entre eles. Desta forma, a codificação do sistema só acontece após a documentação ser revisada e aprovada. As atividades relacionadas à qualidade podem começar a partir do momento em que há documentos sobre o sistema que será desenvolvido.



A principal diferença entre uma metodologia ágil e uma tradicional é que na primeira uma versão mínima do sistema já é entregue a cada ciclo e na segunda o sistema só é entregue quando ele for completamente desenvolvido.

O papel de testador em um time ágil nem sempre é especializado, ou seja, qualquer membro do time de desenvolvimento pode realizar as atividades de teste. No entanto, quando existe um ou mais membros do time que compõem pessoas especializadas em teste de sistema, a chance de realizar testes mais eficientes é bem maior.



Verificação e Validação (V&V) é o nome dado aos processos de verificação e análise que asseguram que o software cumpra com as suas especificações e atenda às necessidades dos clientes que estão pagando por ele



CONCEITOS DE TESTE DE VERIFICAÇÃO

Está relacionada ao fato de estarmos construindo o produto corretamente. Se o software está de acordo com sua especificação

CONCEITOS DE TESTE DE VALIDAÇÃO

A validação tem o objetivo de avaliar se o que foi entregue atende as expectativas do cliente. Ou seja, se os requisitos, independente do que foi planejado, estão sendo implementados para atender a regra de negócio do cliente, se o sistema é realmente aquilo que o cliente quer e está pagando para ter. A validação final do sistema é realizada pelo próprio cliente ou usuário

Para se realizar a verificação e a validação existem duas técnicas distintas:

- Inspeção de Software  são aplicadas em todos os estágios do processo e podem ser complementadas por alguma análise automática de texto dos documentos do sistema.
- Teste de Software  envolvem executar uma implementação do software com os dados de teste e examinar as saídas do teste e seu comportamento operacional.

MÉTODOS ESTRUTURADOS DE VERIFICAÇÃO - REVISÕES

As revisões são métodos de validação de qualidade de um processo ou produto amplamente usado pela equipe técnica do projeto. São consideradas como verdadeiros “filtros” de erros e inconsistências no processo de desenvolvimento de software.

Planejamento para a realização das revisões:

- O que deve ser revisado;
- Quais os resultados esperados;
- Quem deve fazer a revisão;
- Determinar “checkpoints” dentro do ciclo de vida onde a revisão deve ser aplicada;
- Determinar resultados esperados.

Algumas informações importantes no planejamento das revisões são:

- Quem participa?
- Qual informação é requerida antes da revisão?
- Quais pré-condições que devem ser satisfeitas antes que a revisão possa ser conduzida?
- Como Organizar?
- Gerar checklists ou outra indicação do que deve ser coberto na revisão;
- Determinar as condições de término ou critérios que devem ser satisfeitos para que a revisão termine;
- Gerar registros e documentos que devem ser produzidos

As revisões são o principal mecanismo para avaliar o progresso do desenvolvimento de maneira confiável, trazendo vários benefícios para o bom desenvolvimento do software, tais como:

- As revisões trazem à luz as capacidades de cada indivíduo envolvido no desenvolvimento.
- revisões são capazes de revelar lotes ou classes de erros de uma só vez;
- revisões proporcionam retorno já nas primeiras fases, prevenindo que erros mais sérios surjam;
- revisões treinam e educam os participantes
- têm significativo efeito positivo na competência dos desenvolvedores.

Revisões

- As revisões são métodos de validação de qualidade de um processo ou produto amplamente usado pela equipe técnica do projeto.
- São consideradas como verdadeiros “filtros” de erros e inconsistências no processo de desenvolvimento de software.

Tipos de Revisões

- Revisões isoladas
- Revisões técnicas formais
- Reunião de acompanhamento

É muito eficiente na detecção prematura de erros de definições e soluções registradas.

É uma atividade de Garantia de Qualidade de Software

Utilizada na fase de divulgação de documentos

Aplicação e checklist na verificação

Durante o desenvolvimento de software muitos documentos são gerados.

Realizamos verificações para analisarmos esses documentos em cada uma das etapas do desenvolvimento de software é extremamente importante, para garantirmos que problemas não sejam migrados para fases seguintes.

Aplicação e checklist na verificação

Durante a verificação um checklist pode ser útil. Um checklist contém um roteiro de possíveis erros:

Defeitos nos Dados

- Todas as variáveis foram inicializadas?
- As constantes foram nomeadas?
- Tamanho máximo de um vetor?
- Pode haver um overflow de buffer?

Aplicação e checklist na verificação

Defeitos de Controle

- As condições estão corretas?
- Todos os loops estão certos de terminar?
- Os parênteses estão corretos nas condições compostas?
- Num case todas as alternativas estão declaradas?

Aplicação e checklist na verificação

Defeitos Entrada / Saída

- Todas as variáveis de saída têm um valor designado antes de saírem?
- Todas as variáveis de entrada são utilizadas?
- Entradas inesperadas podem corromper os dados?

Aplicação e checklist na verificação

Defeitos de Interface

- Todas as chamadas de funções e métodos têm o número correto de parâmetros.
- Os tipos de parâmetros combinam?
- Os parâmetros estão na ordem certa?

Aplicação e checklist na verificação

Defeito de Gerenciamento de Armazenamento

- Se uma estrutura de link é modificada, todos os links foram corretamente redesignados?
- Se o armazenamento dinâmico é utilizado, o espaço foi alocado corretamente?
- O espaço é explicitamente liberado, depois que não é mais necessário?

Aplicação e checklist na verificação

Defeito de Gerenciamento de Exceções

- Todas as condições de erro foram levantadas?

O checklist do **DoR (Definition of Ready)** é uma lista de critérios que devem ser atendidos antes de um item de trabalho, como uma história de usuário ou uma tarefa, ser considerado pronto para ser incluído em um sprint ou iteração de desenvolvimento. O DoR é uma prática comum em metodologias ágeis, como o Scrum, e visa garantir que os itens de trabalho estejam bem definidos e prontos para serem executados pela equipe de desenvolvimento.

O checklist do DoR pode variar de equipe para equipe e de projeto para projeto, mas geralmente inclui critérios como:

1. **Descrição clara:** O item de trabalho possui uma descrição clara e compreensível que especifica o que deve ser feito.
2. **Critérios de aceitação:** O item de trabalho possui critérios de aceitação bem definidos, que são condições que devem ser atendidas para que o trabalho seja considerado completo.
3. **Estimativa:** O item de trabalho foi estimado pela equipe de desenvolvimento, permitindo que o esforço necessário seja entendido e considerado no planejamento.
4. **Dependências identificadas:** Quaisquer dependências externas necessárias para concluir o item de trabalho foram identificadas e estão disponíveis ou planejadas.

5. **Priorização:** O item de trabalho foi priorizado corretamente em relação aos outros itens do backlog, garantindo que seja abordado no momento adequado.
6. **Recursos disponíveis:** Os recursos necessários para executar o item de trabalho, como conhecimento, ferramentas ou acesso a sistemas, estão disponíveis ou serão fornecidos antes do início do trabalho.
7. **Testabilidade:** O item de trabalho é testável e existem meios para verificar se ele foi implementado corretamente.
8. **Revisão e aprovação:** O item de trabalho foi revisado e aprovado pelo Product Owner ou por outras partes interessadas relevantes, garantindo que esteja alinhado com as necessidades do negócio.

O objetivo do checklist do DoR é garantir que a equipe de desenvolvimento possa iniciar o trabalho sem interrupções desnecessárias, tendo todas as informações e recursos necessários disponíveis. Isso ajuda a melhorar a eficiência do desenvolvimento e a qualidade do produto, uma vez que os itens de trabalho estão em um estado adequado para serem executados.

O checklist do **DoD (Definition of Done)** é uma lista de critérios que devem ser atendidos para que um item de trabalho seja considerado concluído e pronto para ser entregue ao cliente ou ao usuário final. O DoD é uma prática comum em metodologias ágeis, como o Scrum, e tem como objetivo garantir a qualidade e consistência do trabalho realizado pela equipe de desenvolvimento.

O checklist do DoD pode variar de equipe para equipe e de projeto para projeto, mas geralmente inclui critérios como:

- 1. Funcionalidade implementada:** A funcionalidade ou o trabalho proposto foram completamente implementados de acordo com as especificações e requisitos acordados.
- 2. Critérios de aceitação atendidos:** Todos os critérios de aceitação definidos para o item de trabalho foram atendidos. Isso garante que o trabalho esteja em conformidade com as expectativas do cliente ou do usuário final.
- 3. Testes realizados com sucesso:** Os testes necessários foram executados e passaram com sucesso. Isso inclui testes unitários, testes de integração, testes de sistema ou outros tipos de testes relevantes para o item de trabalho.
- 4. Revisão de código concluída:** O código relacionado ao item de trabalho foi revisado por outros membros da equipe e quaisquer problemas ou melhorias identificados foram abordados.

5. **Documentação atualizada:** A documentação relevante, como manuais de usuário, documentação técnica ou qualquer outra documentação relacionada, foi atualizada para refletir as mudanças ou adições feitas no item de trabalho.
6. **Integração realizada:** O trabalho realizado foi integrado com sucesso ao sistema ou ao ambiente em que será implantado. Isso garante que o item de trabalho não cause problemas de compatibilidade ou conflitos com outros componentes do sistema.
7. **Revisão do Product Owner:** O item de trabalho foi revisado e aprovado pelo Product Owner ou por outras partes interessadas relevantes, garantindo que esteja em conformidade com as expectativas do negócio e alinhado com os objetivos do projeto.
8. **Pronto para entrega:** O item de trabalho está pronto para ser entregue ao cliente ou ao usuário final, caso seja necessário.

O **checklist do DoD** é uma ferramenta importante para manter a transparência e a qualidade no processo de desenvolvimento de software. Ele ajuda a garantir que os itens de trabalho sejam concluídos adequadamente e que estejam prontos para serem entregues sem riscos ou problemas significativos. Cada equipe pode personalizar seu checklist do DoD de acordo com suas necessidades e requisitos específicos para garantir uma definição clara de quando um item de trabalho é considerado concluído.

DoR versus DoD

DoR é aplicado antes de um item de trabalho entrar em um sprint, garantindo que ele esteja bem definido e preparado para a execução, enquanto o **DoD** é aplicado ao final do processo de desenvolvimento, garantindo que o trabalho esteja concluído e pronto para entrega. Ambos os checklists são importantes para manter a transparência, a qualidade e o alinhamento durante o desenvolvimento de software.

Cheklis do DoR (Definition of Ready)

- 1- O item cabe em uma Sprint?
- 2- Temos todas as informações necessárias para trabalhar nas tarefas?
- 3- A demanda terá integração com outro órgão?
- 4- A demanda terá integração com outro sistema da CGE?
- 5- A demanda terá alteração ou criação de fluxo?
- 6- Quais os perfis impactados?
- 7- Precisa criar protótipo da tela?
- 8- Foram definidos testes após a implantação ?
- 9- A demanda precisará da equipe da infra?
- 10 - Quais áreas/secretarias serão envolvidas na demanda?

Cheklis do DoD (Definition of Done)

- 1- Contempla os critérios de aceite estabelecidos pela equipe de negócio?
- 2- Foram desenvolvidos todos os testes unitários?
- 3- A demanda foi homologada pela equipe de negócio no ambiente de homologação?
- 4- Foi feita a revisão do código?

Cenário: FESTAS INFANTIS

Rafaela possui vários temas de festas infantis para aluguel.

Ela precisa controlar os aluguéis e para isso quer uma aplicação que permita cadastrar: o nome e o telefone do cliente, o endereço completo da festa, o tema escolhido, a data da festa, a hora de início e término da festa.

Além disso, para alguns clientes antigos, Rafaela oferece descontos. Sendo assim, é preciso saber o valor realmente cobrado num determinado aluguel.

Para cada tema, é preciso controlar: a lista de itens que compõem o tema (ex: castelo, boneca da Cinderela, bruxa etc.), o valor do aluguel e a cor da toalha da mesa que deve ser usada com o tema.

Diferentes Verificações

Os testes de verificação são aplicados respeitando os estágios do desenvolvimento de software:

- Verificação de Negócio.
- Verificação dos Requisitos.
- Verificação de Análise e Modelagem
- Verificação na Implementação

Fases dos Teste de Verificação

Em todas as etapas do processo de desenvolvimento de software são gerados vários documentos.

Para que, chegando no final do desenvolvimento de software tenhamos produzido um produto com qualidade é importante que, durante cada etapa do ciclo os documentos sejam verificados e, após a verificação sejam corrigidos.

Imagine que você vai desenvolver um software e, em cada fase, vai gerando os documentos necessários, porém não verifica se os documentos estão corretos, se o documento necessário a próxima fase foi desenvolvido totalmente.

Fases dos Teste de Verificação

O correto é que as atividades de verificação sejam feitas em cada etapa do processo de desenvolvimento do software.

- Fase de Negócios.
- Fase de Requisitos.
- Fase de Análise e Modelagem.
- Fase de Implementação

Testes de Validação

O processo de qualidade de software está decomposto em fases. Cada fase tem uma sequência de execução dos testes a serem aplicados.

O teste de verificação visa garantir o processo de desenvolvimento de software, enquanto os testes de validação estão focados na garantia da qualidade do produto de software.

Testes de Validação

- Testes de verificação e validação são complementares.
- Em hipótese alguma, estes deverão ser encarados como atividades redundantes.
- Tanto um quanto o outro possui naturezas e objetivos distintos, fortalecendo o processo de detecção de erros e aumentando a qualidade final do produto.

Testes de Validação

Os testes, além de localizar erros, também são necessários para determinar a confiabilidade, o desempenho e para validar a interface com o usuário.



Testes de Validação

- O teste de validação tem o seu foco na aplicação, uma vez que já existe um produto computacional.
- Seu objetivo é identificar o maior número de erros possíveis.
- O sucesso desse teste está apoiada no forte planejamento de todas as atividades de teste.

Testes de Validação

- O foco do estágio de teste do sistema é garantir que se esteja cobrindo todos os requisitos testáveis expressos em um conjunto de casos de teste.
- Isso significa que os critérios de conclusão se concentrarão na cobertura de teste baseada em requisitos.

Testes de Validação

- Nos estágios de teste unitário e de integração, é possível perceber que a cobertura de teste baseada em código é um critério de conclusão mais apropriado.
- Existem basicamente duas estratégias utilizadas para conduzir o processo de validação de software: **testes de caixa branca e testes de caixa preta.**

Testes de Validação – Caixa Branca

- Também conhecido como teste estrutural.
- É aquele que o analista tem total acesso à estrutura interna da entidade (código fonte) sendo analisada e permite, por exemplo, que o analista escolha partes específicas de um componente para serem testadas

Testes de Validação – Caixa Branca

Testes como este permitem ao avaliador concentrar a atenção nos pontos mais importantes ou perigosos do código, de maneira mais precisa do que no caso do teste caixa-preta.

Tais pontos podem até ser identificados durante o desenvolvimento e cobertos com o uso de assertivas e testes.

Testes de Validação – Caixa Branca

Há um elemento em comum entre os testes de caixa branca e caixa preta: em ambos os casos, o avaliador não sabe qual será o comportamento do produto em uma determinada situação.

Testes de Validação – Caixa Preta

O teste de caixa-preta ou teste funcional, também usado no desenvolvimento, no problema de identificação de sistemas, e advém do fato de que ao analisar o comportamento de um objeto, ignora-se totalmente sua construção interna.

O teste de caixa-preta é baseado nos requisitos funcionais do software.

Testes de Validação – Caixa Preta

- Como não há conhecimento sobre a operação interna do programa, o avaliador se concentra nas funções que o software deve desempenhar.
- A partir da especificação são determinadas as saídas esperadas para certos conjuntos de entrada de dados.
- Esse tipo de teste reflete, de certa forma, a óptica do usuário, que está interessado em se servir do programa sem considerar os detalhes de sua construção.

Testes de Validação – Caixa Preta

O teste é particularmente útil para revelar problemas, tais como:

- Funções incorretas ou omitidas;
- Erros de interface;
- Erros de comportamento ou desempenho;
- Erros de iniciação e término

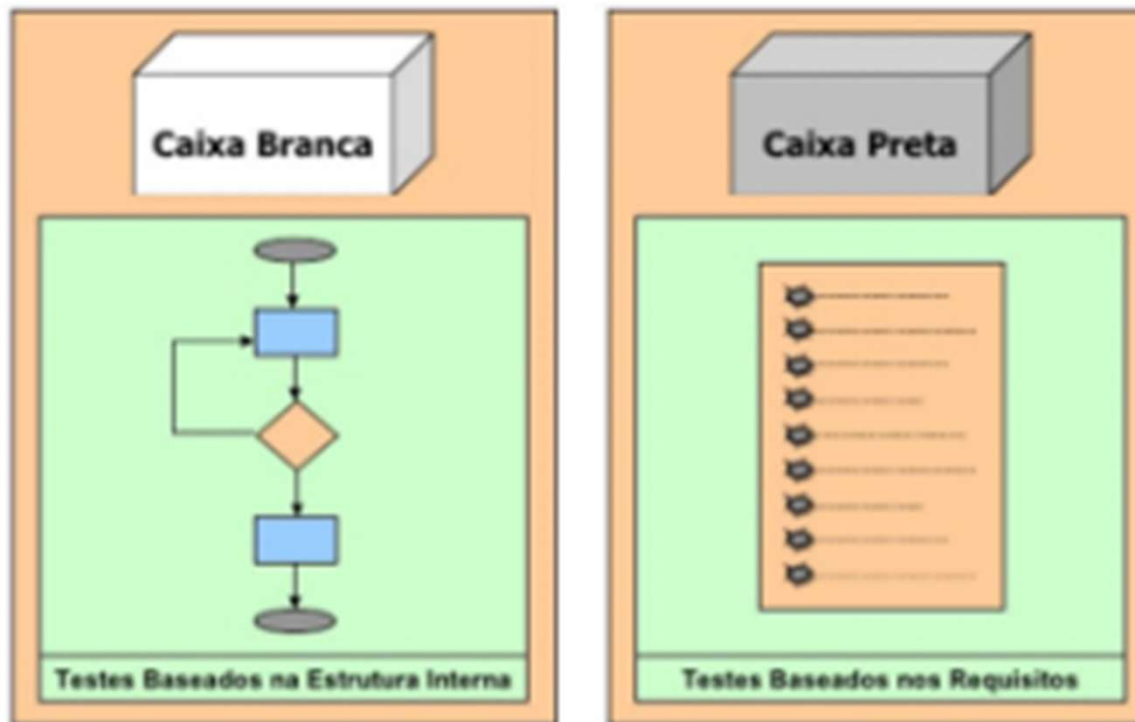
Testes de Validação – Abordagem dos Testes

Abordagens de teste dizem respeito à profundidade da análise a ser realizada.

Em outras palavras, a abordagem de um teste define se o que interessa são apenas as respostas geradas pelo item em teste ou se o seu comportamento interno também deve ser levado em conta.

Testes de Validação – Abordagem dos Testes

Comparação entre abordagens



Testes Progressivos e Regressivos

Os Testes Progressivos são testes elaborados de acordo com a criação de novas funcionalidades do software.

Nesta abordagem é necessário testar somente as inovações do software, assumindo que nenhum erro foi introduzido após seu processo de desenvolvimento.

Testes Progressivos e Regressivos

Já os testes regressivos é uma abordagem que reexecuta um subconjunto (total ou parcial) de testes previamente executados.

Seu objetivo é garantir que uma alteração ou criação de nova funcionalidade não causou impactos nas estruturas já existentes.

Testes Progressivos e Regressivos

Para o **teste regressivo** toda nova versão do produto deveria ser submetida a uma nova sessão de testes, a fim de assegurar que estes novos segmentos não afetaram outras partes do produto.

Categorias de Testes de Validação

Como todo processo são divididos em etapas ou categorias, os testes de validação também possui as suas divisões.

O objetivo de todo teste é detectar os erros existentes no produto de software, porém a realização de todas as categorias de testes descrita a seguir não reflete a realidade vivenciada pelas equipes de teste.

Categorias de Testes de Validação

Testes de Funcionalidade: o direcionamento dos testes funcionais deve ser realizado com base nos documentos de especificação funcional. Este documento descreve o comportamento da aplicação nos diversos cenários existentes para cada requisito de negócio.

A principal característica desta categoria de testes é o grande foco no negócios, com isso, garante-se que não exista diferença entre os requisitos funcionais e o comportamento do software construído.

Categorias de Testes de Validação

Testes de Usabilidade: os testes de Usabilidade têm como meta simular as condições de uso da aplicação sob a perspectiva do usuário. Sendo assim, este tipo de teste tem como foco verificar a facilidade de navegação entre as telas da aplicação, a clareza dos textos e das mensagens apresentadas ao usuário, o volume reduzido de interações para realizar uma determinada função, o padrão visual e etc.

Categorias de Testes de Validação

Testes de Volumes: os testes de volume são caracterizados por determinar os limites de processamento e a carga da aplicação e de toda a sua infraestrutura. Este tipo de teste é conduzido de forma contínua, aumentando o volume das operações realizadas com a aplicação, até que se atinja o limite máximo.

```
import psycopg2
import random
import string
import time

# Configurações de conexão ao banco de dados
host = 'localhost'
port = '5432'
database = 'unifametro'
user = 'postgres'
password = '1234'
```

✓ 0.0s


```
# Função para gerar uma string aleatória
def generate_random_string(length):
    letters = string.ascii_letters
    return ''.join(random.choice(letters) for i in range(length))

def execute_query(query):
    conn = psycopg2.connect(host=host, port=port,
                            database=database, user=user, password=password)
    cursor = conn.cursor()
    cursor.execute(query)
    conn.commit()
    cursor.close()
    conn.close()

# Função para executar uma consulta no banco de dados
def read_query(query):
    conn = psycopg2.connect(host=host, port=port,
                            database=database, user=user, password=password)
    cursor = conn.cursor()
    cursor.execute(query)
    result = cursor.fetchall()
    cursor.close()
    conn.close()
    return result
```

```
CREATE TABLE public.tabela_exemplo  
(  
    id serial,  
    dados text,  
    PRIMARY KEY (id)  
);
```

```
# Função para realizar o teste de volume
def perform_volume_test(num_records):

    # Inserção de registros
    for i in range(num_records):
        dados = generate_random_string(10)
        insert_query = f"INSERT INTO tabela_exemplo (dados) VALUES ('{dados}');"
        execute_query(insert_query)

    # Contagem dos registros
    count_query = "SELECT COUNT(*) FROM tabela_exemplo;"
    result = read_query(count_query)
    count = result[0][0]
    print(f"Número total de registros: {count}")
```

```
# Executa o teste de volume com 1000 registros
start_time = time.time()
perform_volume_test(1000)
end_time = time.time()
elapsed_time = end_time - start_time
print(f"Tempo de execução: {elapsed_time} segundos")
```

✓ 1m 5.5s

Número total de registros: 1000

Tempo de execução: 65.49708414077759 segundos

Relatório de Teste de Volume

Data do Teste: [Data do teste]

Duração do Teste: [Duração total do teste]

Ambiente de Teste: [Descrição do ambiente de teste]

Resumo do Teste

Número total de registros: [Número total de registros]

Tempo médio de inserção: [[Tempo médio de inserção por registro]]

Observações Gerais: [Observações adicionais relevantes]

- 1 – Criar uma segunda tabela
- 2 – Realizar o teste para ambas as tabelas
- 3 – Relatório dever conter dados para 1.000, 10.000, 100.000 registros

Categorias de Testes de Validação

Testes de Carga: os testes de carga servem basicamente para avaliar com uma aplicação e toda sua infraestrutura se comportam quando se aplica um volume de transações superiores aos volumes máximos previstos para esta aplicação e quando se aplica variações bruscas de transações.

```
import psycopg2
import time

# Configurações de conexão ao banco de dados
host = 'localhost'
port = '5432'
database = 'unifametro'
user = 'postgres'
password = '1234'

# Função para executar uma consulta no banco de dados
def execute_query(query):
    conn = psycopg2.connect(host=host, port=port,
                            database=database, user=user, password=password)
    cursor = conn.cursor()
    cursor.execute(query)
    result = cursor.fetchall()
    cursor.close()
    conn.close()
    return result
```



```
# Função para realizar o teste de carga
def perform_load_test(num_queries):
    start_time = time.time()
    for i in range(num_queries):
        # Executa a consulta desejada
        query = "SELECT * FROM tabela_exemplo;"
        result = execute_query(query)
        print(f"Consulta {i+1}: {result}")

    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"\nTempo total de execução: {elapsed_time} segundos")
    print(f"Tempo médio por consulta: {elapsed_time/num_queries} segundos")

# Executa o teste de carga com 10 consultas
perform_load_test(10000)
```


Categorias de Testes de Validação

Testes de Configuração: esta categoria de testes tem por finalidade executar o software sobre inúmeras configurações de software e de hardware. A meta deste teste é garantir que a solução funcione corretamente sobre os diversos ambientes previstos nas fases de levantamento de requisitos.

Categorias de Testes de Validação

Testes de Compatibilidade: esta categoria tem por finalidade executar a aplicação interagindo com as versões anteriores de outras aplicações ou dispositivos móveis.

O objetivo deste teste é garantir que as novas versões de um software estão suportando antigas versões.

Categorias de Testes de Validação

Testes de Segurança: este tipo de teste tem como objetivo encontrar as falhas de segurança que podem comprometer a confidencialidade e a integridade das informações, provocar perdas de dados ou interrupções de processamento.

Categorias de Testes de Validação

Existem ainda outros tipos de testes como:

- Testes de Desempenho
- Testes de Confiabilidade e Disponibilidade
- Testes de Recuperação
- Testes de Instalação

A grande finalidade para esses vários tipos de testes é para garantir a qualidade do produto final.

OBRIGADO!

(85) 3206.6400 (Fortaleza)

(85) 3402.2850 (Maracanaú)

unifametro.edu.br