UNIVERSIDADE SÃO JUDAS TADEU

CIÊNCIA DA COMPUTAÇÃO

GABRIEL FORNICOLA AMORIM - 824148690

GIOVANNI RIBEIRO IANNACE - 82421986

GIOVANNA FONTES DA SILVA – 823148980

LUCAS GASPARETTO NARCIZO DE MORAIS - 82426494

RAPHAEL MIGUEL FOLEGO - 822163593

Conceitos e estratégias de testes de softwares

São Paulo

2025

Conceitos de Teste

É uma Abordagem Fundamental para Garantir a Qualidade de Sistemas. O teste de software é uma das etapas cruciais no ciclo de vida do desenvolvimento de sistemas. Seu objetivo principal é garantir que o software funcione conforme esperado, sem falhas ou defeitos que possam comprometer a experiência do usuário ou a integridade do sistema. Para entender os diferentes aspectos envolvidos nesse processo, é necessário explorar os conceitos de teste, que vão desde a definição dos tipos de teste até as abordagens adotadas pelas equipes de desenvolvimento.

Definição de Teste

De forma geral, o teste de software pode ser definido como o processo de execução de um sistema ou componente com o objetivo de identificar erros ou falhas (ISTQB, 2023). Segundo Beizer (1990), o teste é um "processo de investigação para encontrar defeitos em um sistema, com o objetivo de avaliar sua qualidade e corrigir possíveis problemas antes da liberação para o público".

Para escolher a melhor estratégia de testes de software, é crucial considerar:

- Requisitos do projeto: entender os objetivos, usuários e funcionalidades do software;
- Ciclo de vida do desenvolvimento: alinhar a estratégia com metodologias ágeis ou tradicionais;
- Complexidade do sistema: adaptar a estratégia para sistemas complexos com testes de integração e desempenho;
- Nível de risco: priorizar testes de segurança e regressão para softwares críticos;
- Recursos disponíveis: equilibrar testes automatizados e manuais conforme tempo e equipe;
- Feedback dos usuários: incluir testes de aceitação para garantir alinhamento com as expectativas;

- Documentação e planejamento: detalhar testes, responsabilidades e critérios de aceitação;
- Adaptação contínua: Revisar e ajustar a estratégia conforme o projeto evolui.

As melhores práticas para implementar essas estratégias incluem:

- Planejamento antecipado;
- Automatização de testes;
- Documentação e revisão;
- Feedback e colaboração;
- Priorização por riscos;
- Testes em ambientes reais;
- Integração com DevOps e CI/CD;
- Melhoria contínua.

Verificação

A **verificação** é o processo de avaliar os artefatos do software (como documentos de design, código-fonte e especificações) para assegurar que o produto está sendo desenvolvido corretamente, de acordo com os requisitos especificados. Esse processo é conduzido durante as fases iniciais e intermediárias do desenvolvimento, visando identificar e corrigir erros antes da implementação completa.

Técnicas comuns de verificação incluem:

- **Revisões e inspeções:** Exame minucioso de documentos e código por pares para detectar inconsistências e desvios dos padrões estabelecidos.
- **Análise estática:** Avaliação do código-fonte sem executá-lo, utilizando ferramentas automatizadas para identificar vulnerabilidades, erros de sintaxe e violações de padrões de codificação.

- **Provas formais:** Uso de métodos matemáticos para demonstrar a correção do algoritmo em relação às especificações.

Validação

A **validação**, por outro lado, é o processo de avaliar o produto final para garantir que ele atenda às necessidades e expectativas do usuário final. Isso envolve a execução do software em um ambiente real ou simulado para verificar se ele cumpre sua finalidade pretendida.

Técnicas comuns de validação incluem:

- **Testes funcionais:** Avaliação das funcionalidades do software para assegurar que cada uma opera conforme o esperado.
- **Testes de usabilidade:** Análise da interface e experiência do usuário para garantir que o software seja intuitivo e fácil de usar.
- **Testes de desempenho:** Medição da resposta do software sob diversas condições de carga para verificar sua estabilidade e eficiência.

Importância da Verificação e Validação

A implementação eficaz dos processos de verificação e validação é crucial para:

- **Redução de custos:** Identificar e corrigir erros nas fases iniciais do desenvolvimento é mais econômico do que após a implementação.
- **Melhoria da qualidade:** Assegurar que o software atenda tanto às especificações técnicas quanto às necessidades do usuário final.
- **Aumento da satisfação do cliente:** Entregar um produto confiável e funcional que cumpra ou supere as expectativas do usuário.

Teste unitário

Um teste unitário basicamente é o teste da menor parte testável de um programa.

Se você programa em uma linguagem que suporte paradigma funcional por exemplo, a menor parte testável do seu código deve ser uma função. Então um

teste unitário seria o teste de qualquer função. No caso de orientação a objetos seria o teste de um método de seu objeto.

O que é teste de software

Testes de software são o processo de avaliar e verificar se um produto ou aplicativo de software faz o que deveria fazer. Os benefícios de bons testes incluem a prevenção de bugs e a melhoria do desempenho.

Tipos de teste de software

Existem muitos tipos diferentes de testes de software, cada um com objetivos e estratégias específicos:

- Testes de aceitação: verificar se todo o sistema funciona conforme o esperado.
- Revisão de código: confirmar se o software novo e modificado está seguindo os padrões de codificação de uma organização e segue suas melhores práticas.
- Integration Testing: garantir que os componentes ou funções do software operem juntos.
- Testes de unidades: validar se cada unidade de software funciona conforme o esperado. Uma unidade é o menor componente testável de um aplicativo.
- Testes funcionais: verificar as funções emulando cenários de negócios, com base em requisitos funcionais. O teste de caixa-preta é uma maneira comum de verificar as funções.
- Testes de desempenho: testar como o software é executado sob diferentes cargas de trabalho. Testes de carga, por exemplo, são usados para avaliar o desempenho sob condições de carga reais.
- Testes de regressão: verificar se os novos recursos interrompem ou degradam a funcionalidade. Testes de sanidade podem ser usados para verificar menus, funções e comandos no nível da superfície, quando não há tempo para um teste de regressão completo.

- Testes de segurança: validar se seu software não está aberto a hackers ou outros tipos maliciosos de vulnerabilidades que possam ser explorados para negar acesso aos seus serviços ou fazer com que eles funcionem incorretamente.
- Testes de estresse: testar quanto esforço o sistema pode suportar antes de falhar. Os testes de estresse são considerados um tipo de testes não funcionais.
- Testes de usabilidade: validar com que qualidade um cliente pode usar um sistema ou aplicativo da web para concluir uma tarefa.

Por que os testes de software são importantes:

- Os testes de software são essenciais para garantir a qualidade e confiabilidade de um sistema
- Validam se um trecho de código ou funcionalidade está se comportando conforme o esperado
- São executados continuamente durante o desenvolvimento de software para identificar e corrigir erros

automatização nos testes de software:

A automação de testes de software é o uso de ferramentas para executar, verificar e validar testes de forma automatizada. O objetivo é aumentar a qualidade e eficiência dos testes, além de reduzir a dependência de trabalho manual.

Testes de integração:

O que são testes de integração?

Teste de integração é uma etapa do processo de desenvolvimento de software em que módulos ou componentes são combinados e testados em grupo. Esse tipo de teste visa verificar a eficiência e a segurança da comunicação entre

sistemas. Essa etapa é essencial para garantir que o software funcione sem erros de integração.

O teste de integração é geralmente realizado em fases, começando com os módulos mais simples e progredindo para os módulos mais complexos. Além disso, ele pode ser realizado de forma manual ou automatizada e deve ser feito de forma colaborativa entre desenvolvedores, testadores e engenheiros de software.

Eles podem ser classificados em duas categorias: vertical e horizontal. O teste vertical verifica a interação entre os componentes de um mesmo módulo ou camada, enquanto o teste horizontal testa a interação entre diferentes módulos ou camadas do sistema.

O teste de integração ajuda a identificar e corrigir problemas de interface, que podem surgir quando os módulos de software são combinados. Esses problemas podem causar falhas, erros e desempenho insatisfatório.

- **Quais falhas o teste de integração consegue identificar?**

O teste de integração é uma etapa essencial para garantir a qualidade do software. Ao identificar e corrigir falhas de interface, o teste de integração ajuda a garantir que o sistema funcione corretamente e de forma confiável.

- **Dados perdidos ou corrompidos**

Erros de dados ocorrem quando os dados são corrompidos ou perdidos durante a transferência entre módulos. Isso pode acontecer por vários motivos, como:

- Problemas de comunicação: Os dados podem ser corrompidos durante a transmissão.
- Erros de programação: Os módulos podem não estar usando os dados corretamente.
- Problemas de hardware: Os dispositivos de armazenamento de dados podem estar danificados.

Erros de sintaxe ou semântica nos dados

São erros que ocorrem quando os dados não estão no formato esperado pelo módulo receptor. Esses erros podem ser causados por vários motivos, como:

- Erros de digitação: O módulo transmissor pode digitar os dados incorretamente.
- Erros de programação: O módulo transmissor pode não estar usando o formato de dados correto.
- Problemas de comunicação: Os dados podem ser corrompidos durante a transmissão.

Diferenças nos resultados esperados

São diferenças entre os resultados reais e os resultados esperados de um teste de integração. Essas diferenças podem ser causadas por vários motivos, como:

- Erros de programação: Um módulo pode estar calculando os resultados incorretamente.
- Erros de dados: Os dados que estão sendo usados no teste podem estar incorretos.
- Problemas de comunicação: Os dados podem ser corrompidos durante a transmissão.

Validação de Software

A validação de software é feita por meio de uma série de atividades e testes que visam verificar se o software atende aos requisitos e expectativas dos usuários. Entre as principais etapas da validação de software estão:

1. Planejamento

O planejamento da validação de software envolve a definição dos objetivos, escopo, recursos e cronograma do processo de validação. É nessa etapa que são identificados os requisitos e critérios de aceitação do software.

2. Execução de testes

A execução de testes é uma etapa fundamental da validação de software. Nessa etapa, são realizados testes funcionais, testes de desempenho, testes de segurança e outros tipos de testes para verificar se o software está funcionando corretamente e atendendo aos requisitos estabelecidos.

3. Análise dos resultados

Após a execução dos testes, os resultados são analisados para identificar problemas e falhas no software. Essa análise permite tomar as medidas necessárias para corrigir os problemas identificados e melhorar a qualidade do software.

4. Documentação

A documentação é uma etapa importante da validação de software, pois registra todas as atividades realizadas durante o processo de validação. Isso inclui os resultados dos testes, as correções realizadas e outras informações relevantes.

- **Teste de Software**

Teste de Software é um controle de qualidade, que envolve etapas desde a simulação de uso real até o desenvolvimento de relatórios sobre os resultados obtidos.

O objetivo é verificar se o produto corresponde às funções esperadas e as necessidades dos usuários. O teste de software é uma das últimas etapas antes da disponibilização do aplicativo ao mercado.

São diversos os tipos de testes softwares disponíveis, vamos conhecer um pouco mais sobre eles.

- **Testes de caixa branca**

O profissional pode perceber, por ter acesso ao código fonte, certas etapas do código.

É nesse ponto que é analisado por qual caminho irá ocorrer o fluxo de dados e será conferido se tudo está funcionando da forma esperada.

- **Teste de caixa-preta**

Ao contrário do teste anterior, aqui a pessoa não tem acesso ao código fonte e a estrutura do software. Sendo assim, ele também é chamado de teste funcional.

O teste de caixa-preta pode ser usado para testar casos como:

- Consistir a entrada de datas futuras em datas de nascimento
- Consistir entrada de valores negativos em campos de pagamentos
- Verificar o funcionamento dos botões para prosseguir o fluxo de processamento

Teste de usabilidade e performance

Esse teste tem como objetivo verificar a experiência do usuário. Assim, o responsável irá conferir todo o funcionamento do aplicativo, se o layout está correto, se todos os botões estão funcionando etc. Ele também irá verificar a performance do software para que ele não esteja lento e com um tempo de resposta muito grande.

Esse processo permite também conferir o comportamento do aplicativo em diferentes plataformas e dispositivos, verificar se o layout está responsivo ou não, entre outros.

- **Segurança**

O teste de segurança irá verificar o que diz respeito à proteção contra diversos ataques que o aplicativo pode ser submetido. Além de garantir a proteção dos dados dos clientes que serão inseridos no aplicativo pelo usuário.

- **Manutenção**

Quando um software é produzido ele é feito para durar por muito tempo. Logo, atualizações são constantes e necessárias para aprimorar recursos do programa e a experiência do usuário. Esses testes de manutenção irão conferir se essas atualizações serão aceitas pelo sistema, sem isso, corre o risco de o software ficar desatualizado.

- **Funcional**

Esse teste é a junção do teste de caixa branca e o teste de caixa preta. A importância desse teste é determinar se o software foi programado para fazer o que está fazendo, ou seja, se está cumprindo a sua função.

Todas as funções são testadas de forma diferente, com o intuito de encontrar falhas ou até possíveis melhoramentos no já que está pronto.

- **Depuração**

Depuração corresponde ao processo de localizar e corrigir erros ou bugs no código-fonte de qualquer software. Quando o software não funciona conforme o esperado, os programadores de computadores estudam o código para determinar as causas dos erros. Eles usam ferramentas de depuração para executar o software em um ambiente controlado, verificar o código detalhado, e analisar e corrigir o problema.

O processo de depuração normalmente requer as etapas a seguir.

- **Identificação do erro**

Desenvolvedores, verificadores e usuários finais relatam bugs que descobriram ao testar ou usar o software. Os desenvolvedores devem localizar a linha exata de códigos ou o módulo de código que está causando o bug. Esse pode ser um processo cansativo e demorado.

- **Análise do erro**

Os codificadores analisam o erro registrando todas as alterações de estado do programa e os valores de dados. Eles também priorizam as correções de bugs com base em seu impacto na funcionalidade do software. A equipe de software também identifica um cronograma para correções de bugs, com base nas metas e nos requisitos de desenvolvimento.

- **Correção e validação**

Os desenvolvedores corrigem o bug e executam testes para garantir que o software continua funcionando conforme o esperado. Eles também podem programar novos testes para verificar se o bug se repete no futuro.

Depuração x Teste

A depuração e o teste são processos complementares que garantem que os programas de software estão funcionando como deveriam. Após escrever uma seção completa ou a parte de um código, os programadores realizam testes para identificar bugs e erros. Uma vez que os bugs são encontrados, os codificadores podem iniciar o processo de depuração e trabalhar para solucionar quaisquer erros do software.

Referencias Bibliográficas

[1] https://fullscale.io/blog/software-validation-vs-verification/

[2] https://esr.rnp.br/desenvolvimento-de-sistemas/melhores-praticas-em-testes-de-software/

[3] https://awari.com.br/estrategias-de-teste-de-software-abordagens-e-taticas-em-testes-de-software/

[4] https://www.objective.com.br/insights/estrategia-testes/

[5] https://www.objective.com.br/insights/teste-de-integracao/

[6] https://www.escoladnc.com.br/blog/a-importancia-dos-testes-de-software-conceitos-tipos-ebeneficios/#:~:text=Os%20testes%20de%20software%20s%C3%A3o,identificar%20e%20corrigir%20poss%C3%ADveis%20erros

[7] https://www.ibm.com/br-pt/topics/software-testing

[8] https://dayvsonlima.medium.com/entenda-de-uma-vez-por-todas-o-que-s%C3%A3o-testes-unit%C3%A1rios-para-que-servem-e-como-faz%C3%AA-los-2a6f645bab3

[9] https://www.devmedia.com.br/e-ai-como-voce-testa-seus-codigos/39478

BEIZER, Boris. Software Testing Techniques. 2. ed. New York: Van Nostrand Reinhold, 1990. Unit 1 - pg. 8. Disponível em:

https://acecollege.in/CITS_Upload/Downloads/Books/1077_File.pdf

MYERS, Glenford J. The Art of Software Testing. 2nd ed. Wiley, 2004. Capitulo 2, pg. 5-8. Disponível em : https://web.uettaxila.edu.pk/CMS/seSTbsSp09/notes%5CThe%20Art%20of%20Software%20Testing,%20Second%20Edition.pdf

ISTQB. ISTQB® Certified Tester Foundation Level Syllabus. Versão 4.0. 2023. Pg 14-16 Disponível em: https://bstqb.online/files/syllabus_ctfl_4.0br.pdf