UNIVERSIDADE SÃO JUDAS TADEU

CONCEITOS DE TÉCNICAS DE REVISÃO DE SOFTWARE

GIULIA GABRIELA DE LIMA SANTOS | 823122979
KAUE BRITO VIEIRA | 824211851
KAUÊ DIB DE SOUZA DIAS 823149871
MURILO BONUCCELLI DE OLIVEIRA | 823148988
VICTOR IGNACIO | 823125249
VINÍCIUS SANTANA TEIXERA | 82319112

Conceito de Teste

Os testes de software são fundamentais para garantir a qualidade, segurança e desempenho das aplicações. Eles ajudam a identificar defeitos, prevenir falhas e assegurar que o sistema funcione conforme o esperado. A adoção de boas práticas de testes reduz a incidência de erros em produção, melhora a experiência do usuário e contribui para a manutenção eficiente do código.

Estratégias de Teste de Software

Uma estratégia de teste de software define a abordagem e os métodos usados para garantir a qualidade do sistema. Ela orienta como os testes serão conduzidos, quais tipos serão aplicados e quais ferramentas serão usadas.

Principais Estratégias de Teste de Software

1. Teste Baseado em Requisitos

• Os testes são projetados a partir dos requisitos do sistema. O foco é verificar se todas as funcionalidades exigidas pelo cliente estão implementadas corretamente.

• Exemplo: Se um site de e-commerce exige login para compra, testa-se se o login funciona corretamente.

2. Teste Baseado em Riscos

• Prioriza testes em partes do sistema que apresentam maior risco de falha. Quanto maior o impacto de um erro, mais testes são aplicados nessa área.

• Exemplo: Testar intensivamente um módulo de pagamentos online, pois falhas podem causar prejuízos financeiros.

3. Teste Baseado em Modelos

• Utiliza modelos matemáticos, fluxogramas ou diagramas para criar casos de teste.

• Exemplo: Testes baseados em diagramas de estados para verificar se um sistema de login responde corretamente a diferentes entradas do usuário.

4. Estratégia de Teste Exploratório

• Os testadores exploram o sistema livremente, sem casos de teste predefinidos, procurando falhas e inconsistências.

• Exemplo: Um testador navega por um novo aplicativo sem seguir um roteiro fixo e tenta encontrar falhas inesperadas.

5. Estratégia de Teste Baseada em Automação

• Os testes são automatizados para melhorar a eficiência e a repetibilidade. Essa estratégia é essencial para testes repetitivos ou de grande escala.

• Exemplo: Criar scripts automatizados no Selenium para validar login, carrinho de compras e checkout em um site.

6. Estratégia de Teste Contínuo (CI/CD)

• Os testes são integrados ao pipeline de desenvolvimento para verificar automaticamente cada nova alteração no código. Muito usada em DevOps.

• Exemplo: Sempre que um desenvolvedor faz uma alteração no código, testes automatizados são executados automaticamente para garantir que nada foi quebrado.

Conceitos de Verificação e Validação

• Verificação: Processo de avaliação do software durante o desenvolvimento para garantir que ele atenda às especificações técnicas e requisitos definidos.

• Validação: Processo que ocorre após o desenvolvimento, verificando se o software atende às expectativas e necessidades do usuário final.

A verificação responde à pergunta: “Estamos construindo o software corretamente?”

A validação responde à pergunta: “Estamos construindo o software certo?”

Teste de Software: Teste Unitário e Teste de Integração

Teste Unitário

O teste unitário é a primeira camada de testes no desenvolvimento de software. Ele verifica se uma unidade específica do código, como uma função, método ou classe, funciona corretamente de forma isolada.

Características do Teste Unitário

• Escopo reduzido: Testa uma unidade individual do código sem dependências externas.

• Automatizado: Geralmente escrito com frameworks de teste, como JUnit (Java), pytest (Python) e Jest (JavaScript).

• Executado frequentemente: Deve rodar sempre que houver mudanças no código para garantir que não haja regressões.

Vantagens do Teste Unitário

• Identificação precoce de falhas durante o desenvolvimento.

• Facilitação da refatoração do código.

• Melhoria da qualidade do código e documentação.

Desafios do Teste Unitário

• Não detecta problemas na interação entre componentes.

• Demanda tempo para implementação e manutenção dos testes.

Teste de Integração

O teste de integração tem como objetivo verificar se múltiplos componentes do sistema funcionam corretamente juntos. Ele avalia a comunicação entre módulos, APIs, bancos de dados e outros serviços.

Características do Teste de Integração

• Foca na comunicação entre módulos para garantir que os dados trafeguem corretamente entre diferentes partes do sistema.

• Testa APIs, bancos de dados e microsserviços, permitindo verificar se os sistemas externos e internos se comunicam corretamente.

• Utiliza ferramentas como Postman, Selenium, JUnit e pytest para automação dos testes.

Tipos de Teste de Integração

• Big Bang: Todos os módulos são testados simultaneamente, podendo dificultar a identificação de erros específicos.

• Top-Down: Os módulos de alto nível são testados primeiro, simulando os inferiores com stubs.

• Bottom-Up: Os módulos de baixo nível são testados primeiro, usando drivers para simular os superiores.

• Sanduíche (Hybrid): Combina Top-Down e Bottom-Up para melhor eficiência.

Vantagens do Teste de Integração

• Garante que diferentes partes do sistema funcionem corretamente em conjunto.

• Detecta falhas de comunicação entre módulos.

• Reduz problemas em produção causados por integração mal planejada.

Desafios do Teste de Integração

• Maior complexidade e tempo de execução em comparação aos testes unitários.

• Necessidade de simulação de dependências, como APIs externas ou bancos de dados.

Teste de Validação, Teste de Sistema e Depuração

Teste de Validação

O teste de validação tem como objetivo garantir que o software atende aos requisitos do cliente e funciona conforme esperado no ambiente real. Ele verifica se o produto final corresponde às necessidades do usuário.

Principais Características

• Foco: Avaliação da funcionalidade e da experiência do usuário.

• Quando ocorre: Após o desenvolvimento, antes da entrega ao cliente.

• Métodos comuns: Testes funcionais, testes de aceitação e testes de usabilidade.

Teste de Sistema

O teste de sistema verifica o software como um todo, garantindo que todos os módulos funcionam corretamente quando integrados. Ele simula o ambiente real de uso, avaliando funcionalidade, desempenho e segurança.

Principais Características

• Foco: Avaliação global do sistema.

• Quando ocorre: Após os testes de unidade e integração, antes do teste de validação.

• Métodos comuns: Testes funcionais, de desempenho, de carga e de segurança.

Depuração

A depuração é o processo de encontrar, analisar e corrigir erros (bugs) no código-fonte. Diferente do teste de software, que identifica falhas, a depuração investiga e resolve essas falhas.

Principais Características

• Foco: Identificação e correção de bugs no código.

• Quando ocorre: Durante todo o desenvolvimento e testes.

• Métodos comuns: Uso de logs, breakpoints e ferramentas de depuração.

• Quem realiza: Desenvolvedores.

Conclusão

A realização de testes de software é essencial para garantir a qualidade e confiabilidade de um sistema. Diferentes tipos de testes desempenham papéis complementares: testes unitários asseguram a funcionalidade de pequenos componentes, testes de integração verificam a comunicação entre módulos e testes de sistema analisam o comportamento do software como um todo. Além disso, a validação e a depuração são etapas cruciais para a entrega de um produto robusto e seguro. A escolha da estratégia de testes ideal deve levar em conta os requisitos do projeto, os riscos envolvidos e a necessidade de automação, garantindo um desenvolvimento eficiente e livre de falhas.