

Aula 5

Teste de Software

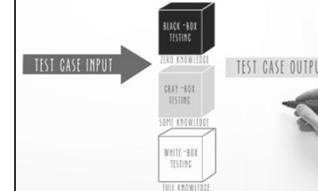
Profª Maristela Weinfurter

Conversa Inicial



Stock Studio / Shutterstock

TYPES OF PENETRATION TESTING



astel design / Shutterstock

Tipos de testes - Caixa branca

- O teste de caixa branca baseado em fluxo de controle é a maneira mais barata de encontrar muitos tipos de bugs de codificação porque muitos deles são erros em fluxos de controle que existem inteiramente dentro de uma linha ou algumas linhas de uma unidade de código

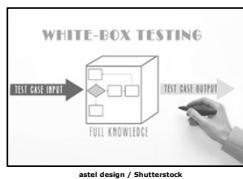
Tipos de testes - Caixa branca

- Esse tipo de teste ainda inclui verificações de unidade, de integração, validação das estruturas e outros testes para validação da qualidade da funcionalidade do código
- A equipe de teste pode usar ferramentas de análise estática de código, bem como ferramentas de depuração, para identificar problemas de performance, segurança e correção de bugs

Tipos de testes - Caixa branca

- Cobertura da declaração
- Cobertura de ramo (ou decisão)
- Cobertura de condição
- Cobertura multicondição
- Cobertura de decisão multicondição
- Cobertura de loop
- Cobertura do caminho

- Para que realmente implementemos o teste de caixa branca, é importante que as entradas, resultados e especificações estejam bem descritas e elaborados, bem como todo o critério de aceite para certificação da qualidade do código



Tipos de testes - Mudanças

Tipos de testes - Mudanças

- Testes relacionados à mudança são realizados para verificação de mudanças em um sistema ou aplicativo que podem afetar outras partes do sistema de maneira indesejada. Eles são executados para garantir a integridade do sistema e minimizar o risco de problemas em áreas não relacionadas à versão original

Tipos de testes - Mudanças

- Esse tipo de teste pode ser realizado após as mudanças implementadas, para as quais o grande objetivo concentra-se nos possíveis efeitos indesejados que as alterações possam causar em qualquer parte não modificada. Pode ainda incluir testes de unidade, integração, regressão e outros tipos de testes que visam garantir que o sistema continue funcionando de maneira eficiente e correta após a mudança

- O que importa num teste relacionado a mudanças é que ele continue garantindo a qualidade do nosso software
- Inevitavelmente estaremos trabalhando em testes estruturais (caixa branca), testes funcionais (caixa preta), testes não funcionais e, finalmente, na sua completude, testes relacionados à mudança



Níveis e tipos de testes

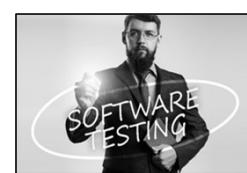
- As atividades de teste relacionadas a um determinado nível da arquitetura são conhecidas como um “nível” de teste, e cada nível de teste é uma única instância do processo de teste

- Teste de componente
- Base de teste
- Objetos de teste
- Verificação da funcionalidade interna
- Teste de desenvolvedor
- Teste X depuração

Níveis e tipos de testes

- Funcionalidade de teste
- Teste de eficiência
- Teste de manutenibilidade
- Estratégias de teste
- Testes de caixa branca
- Teste em primeiro lugar
- Teste de integração

- A eficiência do teste é medida usando a relação entre os custos do teste (equipe, ferramentas e assim por diante) e a utilidade (o número e a gravidade das falhas descobertas) para um determinado nível de teste
- O gerente de teste é responsável por escolher e implementar a estratégia ideal de teste e integração para o projeto em questão



Testes não funcionais

Testes não funcionais

- Os testes não funcionais verificam várias características de um software, e não precisamos compreender as regras de negócio
- Tais testes complementam os testes funcionais e outros

Testes não funcionais

- **Carga:** medição do comportamento do sistema sob carga crescente (por exemplo, número de usuários paralelos, número de transações)
- **Desempenho:** medição da velocidade de processamento e tempo de resposta para casos de uso específicos, geralmente em conjunto com o aumento da carga

Testes não funcionais

- **Volume de dados:** observação do comportamento do sistema dependente de volumes de dados (por exemplo, ao processar arquivos muito grandes)
- **Estresse:** observação do comportamento do sistema em situações de sobrecarga

Testes não funcionais

- **Segurança de dados:** combate ao sistema não autorizado e/ou acesso a dados
- **Estabilidade/confiabilidade:** em uso constante (por exemplo, medindo o número de falhas do sistema por hora para perfis de usuário específicos)

Testes não funcionais

- **Robustez:** quando submetido a erros do usuário, erros de programação, falha de hardware e similares. Testando o tratamento de exceções e o comportamento de reinicialização/recuperação
- **Compatibilidade/conversão de dados:** testar a compatibilidade com sistemas existentes, especialmente durante a importação/exportação de dados

Testes não funcionais

- Diferentes configurações: por exemplo, usando diferentes versões de sistema operacional, idiomas ou plataformas de hardware
- Usabilidade: testar a facilidade de aprendizado e a simplicidade de uso, incluindo a comprehensibilidade da saída para vários grupos de usuários

Testes não funcionais

- Conformidade da documentação do sistema com o comportamento do sistema: por exemplo, manual do usuário versus GUI ou descrição da configuração versus comportamento real
- Manutenibilidade: comprehensibilidade e atualização da documentação de desenvolvimento, estrutura modular e assim por diante

- Os requisitos não funcionais são frequentemente formulados de forma incompleta ou vaga. Atributos como: "O sistema precisa ser fácil de usar" ou "resposta rápida" não podem ser testados em sua forma atual



Teste de manutenção

Teste de manutenção

- A manutenibilidade incorpora todos os atributos que influenciam quão fácil (ou difícil) é aprimorar ou estender um programa. O fator crítico aqui é a quantidade de esforço necessária para um desenvolvedor (equipe) obter uma compreensão do programa existente e seu contexto

Teste de manutenção

- Os principais aspectos de manutenibilidade que precisam ser testados são estrutura de código, modularidade, comentários de código, comprehensibilidade e atualização da documentação e assim por diante

Teste de manutenção

- **SUT (System Under Test - Sistema em Teste)**
 - **Controlando o SUT**
 - ✓ Os testes automatizados usam interfaces para acionar ações e eventos dentro do SUT. Isso é feito, por exemplo, por meio de APIs, protocolos de comunicação, elementos de interface do usuário ou comutadores eletrônicos

Teste de manutenção

- **SUT (System Under Test - Sistema em Teste)**
 - **Observabilidade**
 - ✓ Testes automatizados usam interfaces para verificar se o comportamento real do SUT corresponde ao comportamento esperado

Teste de manutenção

- **SUT (System Under Test - Sistema em Teste)**
 - **Arquitetura clara**
 - ✓ Para obter uma estratégia de automação de teste consistente e transparente, você precisa definir claramente quais interfaces estão disponíveis para automação de teste em qual nível de teste. O grau de intrusão e os efeitos da ferramenta de automação no SUT também devem ser avaliados

- A manutenção de software é algo inevitável, até porque um software é algo dinâmico, que sofre mudanças constantes, seja por melhorias, seja por correções



Semanche / Shutterstock