Aluno: Diego Carlito Rodrigues de Souza
Matrícula: 221007690
Professora: Elaine Venson
Turma: 01

Effective Software Testing

1 Effective and systematic software testing

1. **Responsabilidade dos Desenvolvedores**: Os desenvolvedores não são mais apenas responsáveis por escrever código, mas também pela qualidade dos sistemas de software que produzem. Isso destaca a importância do teste de software como parte integrante do processo de desenvolvimento.
2. **Processo de Escrita de Testes**: O processo de escrever testes é apresentado como uma maneira de entender melhor o que os programas precisam fazer e obter feedback sobre o design do código. Destaca-se a importância de prestar atenção à qualidade do código de teste para garantir a manutenção e evolução suave do conjunto de testes.
3. **Técnicas de Teste**: Diversas técnicas de teste são discutidas, incluindo o desenvolvimento orientado a testes (TDD), testes baseados em propriedades e testes baseados em exemplos. Essas técnicas fornecem abordagens diferentes para garantir a qualidade do software.
4. **Automação de Testes**: A automação de testes é destacada como crucial para um processo de teste eficaz, permitindo a execução rápida e repetível de testes, o que é essencial para a entrega rápida e confiável de software.
5. **Princípios de Teste de Software**: São apresentados vários princípios de teste de software, incluindo a impossibilidade de testes exaustivos, a importância da escolha criteriosa dos casos de teste e a compreensão de que nenhum teste é perfeito.
6. **Pirâmide de Testes**: A pirâmide de testes é introduzida como uma maneira de visualizar a distribuição ideal de diferentes tipos de testes, destacando a importância dos testes de unidade, integração e sistema, com vantagens e desvantagens de cada nível.

The Art of Software Testing

A Self-Assessment Test

1. **Complexidade Crescente:** O teste de software se tornou mais desafiador devido à diversidade de linguagens de programação, sistemas operacionais e plataformas de hardware disponíveis atualmente.
2. **Facilidades Atuais:** Por outro lado, o teste também se tornou mais fácil em alguns aspectos devido ao avanço das linguagens de programação, sistemas operacionais e plataformas de hardware, que oferecem rotinas intrínsecas e bem testadas que podem ser incorporadas em aplicativos sem a necessidade de desenvolvê-las do zero.
3. **Ferramentas de Desenvolvimento Melhores:** O texto menciona a existência de ferramentas de desenvolvimento melhores, que ajudam no processo de teste de software.
4. **Pressão por Prazos Apertados:** A pressão por prazos apertados em um ambiente de desenvolvimento cada vez mais complexo pode levar à evitação de protocolos de teste mais complexos, focando apenas nos testes mais óbvios.
5. **Definição de Teste de Software:** O teste de software é descrito como um processo ou série de processos destinados a garantir que o código de computador realize o que foi projetado para fazer e, ao mesmo tempo, que não realize nada de não intencionado. O objetivo é que o software seja previsível, consistente e não apresente surpresas para os usuários.

The Psychology and Economics of Software Testing

1- O que é teste de software?

O teste de software é um processo crucial no desenvolvimento de software que visa garantir a qualidade e confiabilidade do programa. Através da execução do programa em diferentes cenários e condições, o teste busca identificar e corrigir erros, garantindo que o software atenda às suas especificações e funcione conforme o esperado pelos usuários.

2- Importância do teste de software:

- **Qualidade:** O teste de software identifica e corrige erros que podem afetar a funcionalidade, desempenho, segurança e usabilidade do programa. Isso resulta em um software mais confiável e robusto, proporcionando uma melhor experiência ao usuário.
- **Confiança:** O teste de software aumenta a confiança dos usuários no programa, pois garante que ele foi testado e validado em diferentes cenários. Isso é especialmente importante para softwares críticos, como sistemas bancários e médicos, onde falhas podem ter consequências graves.

- **Economia:** O teste de software pode ajudar a prevenir custos futuros associados à correção de erros em campo. Detectar e corrigir erros em um estágio inicial do desenvolvimento é muito mais barato do que fazê-lo após o lançamento do software.

Estratégias de teste:

3- Caixa preta:

- **Foco:** Comportamento externo do programa.
- **Dados de teste:** Baseados nas especificações do programa.
- **Técnicas:** Teste de unidade, teste de integração, teste de sistema, teste de aceitação.
- **Vantagens:**
  - Fácil de implementar.
  - Não requer conhecimento da estrutura interna do programa.
- **Desvantagens:**
  - Pode não ser tão eficaz na detecção de erros de lógica interna.
  - Pode ser difícil criar dados de teste que cubram todos os cenários possíveis.

4- Caixa branca:

- **Foco:** Estrutura interna do programa.
- **Dados de teste:** Baseados na lógica do programa.
- **Técnicas:** Teste de unidade, teste de cobertura de código, teste de mutação.
- **Vantagens:**
  - Pode ser mais eficaz na detecção de erros de lógica interna.
  - Permite um teste mais direcionado e eficiente.
- **Desvantagens:**
  - Pode ser mais difícil de implementar.
  - Requer conhecimento da estrutura interna do programa.

5- Princípios do teste de software:

1. Definição do resultado esperado:

- Cada teste deve ter um resultado esperado definido para que se possa determinar se o teste foi bem-sucedido ou não.
- O resultado esperado deve ser claro, conciso e verificável.

2. Evitar que programadores testem seus próprios programas:

- Os programadores podem estar muito próximos do programa para serem objetivos na avaliação dos resultados dos testes.
- É recomendável que os testes sejam realizados por uma equipe independente de testadores.

3. A organização não deve testar seus próprios programas:

- Similar ao princípio 2, a organização pode ter um viés na avaliação dos resultados dos testes.
- É recomendável que os testes sejam realizados por uma equipe externa independente.

4. Inspeção dos resultados de cada teste:

- É importante analisar cuidadosamente os resultados de cada teste para identificar falhas e possíveis problemas.
- A análise dos resultados pode ajudar a identificar a causa dos erros e prevenir sua recorrência.

5. Teste de entradas válidas e inválidas:

- O teste deve considerar tanto entradas válidas quanto inválidas para garantir que o programa se comporte de forma adequada em todos os cenários.
- Testes com entradas inválidas podem ajudar a identificar vulnerabilidades e falhas de segurança.

6. Verificação do que o programa faz e não faz:

- O teste deve verificar não apenas o que o programa faz, mas também o que ele não faz.
- Isso é importante para garantir que o programa não se comporte de forma inesperada ou indesejada.

7. Evitar testes descartáveis:

- Os testes devem ser reutilizáveis sempre que possível para evitar retrabalho e desperdício de tempo.
- A criação de testes automatizados pode ajudar a tornar os testes mais eficientes e reutilizáveis.

8. Não presumir que não haverá erros:

- É importante ter uma atitude realista e esperar que o programa contenha erros.
- O teste deve ser realizado com o objetivo de encontrar erros, não de provar que o programa está livre de erros.

9. Probabilidade de erros em seções com muitos erros:

- Seções do programa com muitos erros tendem a ter uma maior probabilidade de conter outros erros.
- É recomendável dedicar mais atenção ao teste dessas seções.

10. Teste como tarefa criativa e intelectualmente desafiadora