Felipe dos Santos Gamboa - 821137220

Estratégias de Testes de Software

No desenvolvimento de software, uma das etapas cruciais para garantir a qualidade e confiabilidade de uma aplicação é a fase de testes. As estratégias de testes de software são planejadas para identificar defeitos, verificar o comportamento esperado do sistema e garantir que ele esteja alinhado com os requisitos estabelecidos. Existem várias abordagens para a realização de testes, cada uma com seus propósitos específicos e momentos apropriados para aplicação.

1. Testes Funcionais

Os testes funcionais avaliam se o software cumpre suas funções conforme os requisitos especificados. Nessa parte, os casos de teste são desenvolvidos para verificar a interação entre as entradas e saídas do sistema, garantindo que os resultados obtidos estejam de acordo com o comportamento esperado.

Exemplos de Testes Funcionais:

- Teste de interface do usuário (UI);
- Testes de integração do sistema;
- Testes de aceitação pelo usuário (UAT).

2. Testes Não Funcionais

Os testes não funcionais verificam atributos de qualidade do software, como desempenho, segurança, usabilidade e confiabilidade. Essas características, mesmo que não estejam diretamente relacionadas à funcionalidade do sistema, são essenciais para sua eficiência como um todo.

Exemplos de Testes Não Funcionais:

- Testes de performance (tempo de resposta, carga);
- Testes de segurança (ataques e vulnerabilidades);
- Testes de compatibilidade entre plataformas e dispositivos.

3. Testes de Caixa Branca (White-Box)

Esse tipo de teste examina a estrutura interna do software, com o foco em verificar o funcionamento correto do código-fonte. Os testadores precisam entender a lógica interna do sistema e escrever testes que cobrem todas as possíveis ramificações do código.

Técnicas utilizadas:

- Cobertura de código (analisando o percentual de código testado);
- Testes unitários (testes de pequenas partes do código, como métodos ou funções).

4. Testes de Caixa Preta (Black-Box)

Nos testes de caixa preta, o testador não tem conhecimento do código-fonte. A abordagem é baseada nas entradas e saídas do sistema, verificando se o software se comporta de acordo com as especificações sem considerar sua implementação interna.Com esse método, temos a vantagem da simplicidade na criação de casos de teste, sendo ideal para testar funções complexas sem precisar conhecer o código.

5. Testes de Regressão

Os testes de regressão garantem que as novas alterações no código não quebrem funcionalidades que já estavam funcionando corretamente. Toda vez que uma modificação é feita, seja ela um bug fix ou uma nova feature, é necessário garantir que o comportamento anterior do sistema permaneça intacto. As ferramentas mais comuns para esses testes são Selenium, JUnit e TestNG.

6. Testes Automatizados

A automação de testes busca agilizar o processo de validação, especialmente em casos onde existem muitos testes repetitivos. Os testes automatizados são geralmente usados para tarefas como testes de regressão, performance e testes funcionais. As vantagens de realizar o teste automatizado seriam a redução do tempo de execução dos testes, uma maior precisão na repetição de testes e a cobertura de um grande número de cenários.

7. Testes Exploratório

Os testes exploratórios são realizados sem um plano formal de testes. Aqui, o foco é dado ao aprendizado sobre o sistema e à investigação de potenciais falhas. Os testadores exploram livremente o software, tentando descobrir comportamentos inesperados.

Conclusão

A adoção de estratégias de testes adequadas é fundamental para garantir a qualidade e o sucesso de um projeto de software. A combinação de diferentes tipos de testes (funcionais, não funcionais, automatizados e etc.) permite uma cobertura mais ampla e eficiente das possíveis falhas, assegurando que o sistema funcione corretamente em ambientes de produção.