ENG
SOFT - TESTES

TESTES DE SOFTWARE

1. Fundamentos sobre testes de software

A atividade de teste de software sempre foi considerada como um gasto de tempo desnecessário, uma atividade de segunda classe, uma vez que a programação é a atividade principal no desenvolvimento de software. Os testes tinham apenas o objetivo de mostrar que o sistema funcionava. Somente a partir da década de 1980 que começaram a surgir métodos de testes que passaram a fazer parte do processo de desenvolvimento de software, de maneira formal e como uma atividade essencial ao processo de construção, que passa a ter o objetivo de garantir que o sistema atende aos requisitos especificados. No final da década de 1990 começam a surgir as funções de gerente de testes, analistas de testes e operadores de testes que são especialistas no planejamento, elaboração e execução dos testes tornando os testes cada vez mais relevante no ciclo de vida do software e assumindo um perfil de

prevenção de problemas, e não apenas para encontrar erros. Atualmente, o grau de exigência dos usuários com a qualidade dos produtos de software está cada vez maior, a complexidade das aplicações em relação a granularidade, a segurança e a integração com outros sistemas e o aumento da concorrência no mercado de software tornam os testes uma atividade essencial e indispensável, com equipes exclusivas e dedicadas a essas ações com o objetivo de garantir e controlar a qualidade do sistema.

2. Conceitos de testes de software

As definições sobre testes de software variam, mas todas convergem para os conceitos básicos de encontrar defeitos em um software que está sendo testado, se está de acordo com os requisitos definidos pelo usuário e se faz o que deveria ser feito.

Segundo Myers (1979), testar um software é um processo de executar um programa ou sistema com a intenção de encontrar defeitos. Para Dijkstra (1985), os testes podem mostrar a presença de falhas em um software, mas nunca a sua ausência. Para Hetzel (1988), testes é uma atividade que, a partir da avaliação de um atributo ou capacidade de um programa, seja possível determinar se ele alcança os resultados esperados. Para o IEEE, testes é um processo de execução de um sistema ou programa, sob condições especificas, para detectar diferenças entre os resultados obtidos e os esperados.
É importante ressaltar que muitos desenvolvedores ainda confundem o processo de testes com o processo de depuração de um programa, mas ambos são completamente diferentes, pois enquanto os testes é uma busca estruturada para encontrar erros em um programa pronto, a depuração, mais conhecida como "debug", é um processo de busca de erros de forma não estruturada durante a execução de um programa, através de uma ferramenta de apoio ao desenvolvimento, onde uma

vez encontrado, o erro deve ser corrigido.
A depuração, normalmente, ocorre durante a atividade de programação e antes da execução dos testes propriamente dito. Os testes devem ser realizados pelos desenvolvedores através dos testes unitários, pelos testadores através dos testes integrados guiados por roteiro de testes elaborados a partir da especificação inicial e pelos usuários através dos testes de aceitação, no qual é verificado se o software atende às suas necessidades. Ou seja, testes de software é uma atividade de validação e é responsabilidade de todos que participam do processo de desenvolvimento para garantir a qualidade.

3. Por que devemos testar um software?

Além do fator qualidade, que foi exposto até aqui, que afeta a satisfação do cliente e é um dos principais indicadores de uma organização, o fator custo também é um grande incentivador para que as organizações apliquem e utilizem o processo de testes no ciclo de desenvolvimento

de software. Segundo um dos maiores especialistas em desenvolvimento de software no mundo, Boehm (1976), quanto mais tarde um defeito for identificado, mais caro fica para corrigi-lo. Além disso, os custos para se identificar e corrigir os defeitos em um software aumentam exponencialmente na proporção que o trabalho evolui em suas fases de desenvolvimento.

De acordo com Myers (1979), quanto mais cedo descoberto e corrigido o defeito, menor é o seu custo para o projeto, pois esse custo cresce em até 10 (dez) vezes para cada fase que o projeto do software avança. Essa afirmação é conhecida como a Regra 10 de Myers. É do conhecimento do mercado de tecnologia da informação que as organizações chegam a gastar até 30% (trinta por cento) do esforço total do desenvolvimento de software realizando testes e, no caso de construção de softwares críticos, como software de controle de voo, trens e navios, a fase de testes pode custar de 3 a 5 vezes mais que todas as demais fases de um projeto tradicional. Garantir que o software esteja sem erros e defeitos é um dos desafios da engenharia de

software, mas não é possível se rever todas as combinações de testes que garantam a total cobertura de testes em uma aplicação. Portanto, sempre haverá algo a se fazer, seja testar, seja para corrigir um defeito. Mas, de todas as formas de verificação e validação da qualidade, o processo de testes é forma mais eficaz de se aproximar do erro zero e, aliada ao trabalho de equipes de testes cada vez mais especializadas e independentes, os resultados tendem a ficar cada vez melhores e o objetivo de aumento da qualidade e da redução dos custos podem ser alcançados com esforços cada vez menores.

4. Tipos de Testes

Normalmente, a preocupação dos desenvolvedores está em realizar testes que garantam que o software atende as necessidades dos usuários. Porém, os testes não se limitam apenas aos requisitos funcionais. Os requisitos funcionais envolvem o correto funcionamento do software, sua integração com outros sistemas, as suas interfaces e a garantia

de que qualquer alteração feita não afeta a aplicação. Os requisitos não funcionais surgem principalmente na fase de projeto arquitetural e/ou detalhado, estão ligados às características comportamentais do software e não são solicitadas pelos usuários. Esses requisitos devem ser avaliados pela equipe de desenvolvimento, validados junto aos usuários e definidos aqueles que devem ser aplicados a cada tipo de sistema. Nesse tópico são apresentados e descritos os diversos tipos de testes mais utilizados pelo mercado de tecnologia da informação, tanto para os requisitos funcionais, como para os requisitos não funcionais e que devem ser parte integrante do checklist de qualidade de desenvolvimento de software, principalmente para a fase de projeto.

4.1 Tipos de testes para requisitos funcionais

-Testes de regressão

Tem como objetivo garantir que, mesmo após mudanças realizadas nas fases de desenvolvimento

ou manutenção, a aplicação continua funcionando adequadamente e produzindo os resultados esperados.

-Testes de Interoperabilidade

Visam avaliar e garantir que a comunicação entre os sistemas envolvidos e entre os ambientes computacionais da aplicação estão funcionando adequadamente. Por exemplo: Ao acessar um sistema de validação de CEP, o mesmo retorna valores válidos.

-Testes Alfa e Beta

São testes executados após a aplicação estar pronta e feita pelos usuários finais da aplicação. Os roteiros de testes não são seguidos e o objetivo é identificar o maior número de defeitos possíveis antes de liberar a aplicação para a produção.

-Testes de usabilidade

Avalia a facilidade de uso da aplicação quanto a visão dos dados na tela, cores, disposição dos componentes de interação dos usuários, facilidade de interpretação das mensagens e o conteúdo e clareza das informações nas telas.

4.2 Tipos de testes para requisitos não funcionais

-Testes de carga ou stress

São testes que tem como objetivo avaliar o comportamento da aplicação sob condições extremas de acessos simultâneos ou de requisições ao servidor para verificar se suporta o volume esperado. Devido ao volume de requisições que precisam ser gerados, esses testes são realizados por ferramentas específicas para esse fim. Exemplo: a aplicação deve suportar até mil usuários realizando "login" ao mesmo tempo ou responder até 300 requisições por minuto.

-Testes de desempenho ou performance

Tem por objetivo verificar se a aplicação responde as requisições dos usuários dentro do tempo esperado. Também são testes automatizados. Por exemplo: a cada solicitação do usuário o sistema deve responder em no máximo 5 segundos.

-Testes de segurança

Verifica a capacidade da aplicação em lidar com as tentativas não autorizadas de acesso e aos acessos dos usuários autorizados.

-Testes de recuperação ou disaster recovery

O objetivo desses testes é validar como a aplicação se comporta em situações extremas de falta de energia, queda da rede, falha de comunicação, entre outras.

-Testes de portabilidade

Visa avaliar a instalação e o funcionamento da aplicação em ambientes operacionais diferentes, previstos no planejamento.

-Testes de Confiabilidade ou Disponibilidade

Visa avaliar o comportamento da aplicação quando submetido a falha de algum item do seu ambiente operacional, como servidor de aplicação, servidor de banco de dados, entre outros. Nessa situação, o software deverá manter-se funcionando parcialmente e/ou tratar as falhas adequadamente através de mensagens aos usuários.

4.3 Tipos de testes para ambiente Testes Estáticos

Validam se as especificações, diagramas de casos de uso, diagrama de classes, modelo de dados, entre outros, estão coerentes com a aplicação

desenvolvida. São realizados através de revisões técnicas formais ou inspeções.

-Testes de aderência de código

Visa avaliar se o código-fonte da aplicação está de acordo com os padrões e melhores práticas previstas no processo de qualidade. São executados através de ferramentas ou através de inspeções.

-Testes de configuração

Avaliam se o software se comporta adequadamente de acordo com as especificações de hardware e software planejadas. Por exemplo: o software funciona nas versões x e y do browser Z?

-Testes de navegação

É avaliado o comportamento da aplicação no que tange a navegação entre telas, links para outros sistemas, entre outros.

-Testes de instalação

Verifica se o plano de instalação do sistema está correto e se a aplicação funciona após a instalação em um novo ambiente.