Visão introdutória sobre a funcionalidade do teste de softwre no desenvolvimento de sistemas

Jenifer de Abreu
Joseneuza Julita Pimenta de Aguiar
Rogério Oliveira da Silva

Resumo: Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. O conceito de teste de software pode ser compreendido através de uma visão intuitiva ou mesmo de uma maneira formal. Existem atualmente várias definições para esse conceito. De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu comportamento corre de acordo com o especificado. O objetivo principal desta tarefa é revelar o número máximo de falhas dispondo do mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos. Ao longo deste artigo, iremos discutir os principais conceitos relacionados às atividades de teste, as principais técnicas e critérios de teste que podem ser utilizados para verificação ou validação de um produto, assim como o ciclo de vida dos testes e os papeis e responsabilidades.

Palavras-Chave: Teste de Software. Qualidade de Software. Desenvolvimento de Software.

Abstract: Software testing is the process of running a product to determine if it has met its specifications and has worked correctly in the environment for which it was designed. Its purpose is to disclose defects in a product, so that the causes of those failures are identified and can be corrected by the development team before final delivery. The concept of software testing can be understood through intuitive insight or even in a formal way. There are currently several definitions for this concept. In a simple way, testing a software means checking through a controlled run if its behavior runs according to the specified. The main objective of this task is to reveal the maximum number of failures with the minimum of effort, that is, to show those who develop if the results are or not according to the established standards. Throughout this article, we will discuss the main concepts related to the test activities, the main techniques and test criteria that can be used to verify or validate a product, as well as the life cycle of the tests and the roles and responsibilities.

Keywords: Software Testing. Software quality. Software development.

1. Introdução

Atualmente, cada vez mais os softwares estão presentes na vida das pessoas, seja em celulares, bancos, aviões, entre outros. Conseqüentemente os mesmos estão se tornando cada vez mais complexos, devido ao surgimento de novas tecnologias. A maioria das pessoas passam por alguma experiência com um

software que não funciona como o esperado, softwares que não funcionam corretamente podem levar a muitos problemas para os usuários.

Para evitar problemas é necessário investir em testes, afinal, testes em sistemas e em documentações reduzem os riscos da ocorrência de defeitos do software no ambiente de produção onde são encontrados pelo cliente, contribuindo assim para a qualidade dos sistemas, pois quanto mais cedo os defeitos forem encontrados, antes da implantação do sistema, o custo de correção é menor em relação ao encontrado da fase de produção afirma a regra 10 de Myers.

O grande desafio das empresas é produzir softwares com qualidade, em um curto espaço de tempo, com baixo custo e atender as expectativas do cliente com o produto desenvolvido. Os testes têm por finalidade agregar qualidade ao produto podendo também fazer uma medição desta qualidade em relação aos defeitos encontrados, pois, caso sejam encontrados poucos defeitos, o software será mais confiável, com os testes é possível também antecipar a descoberta de falhas e incompatibilidades, reduzindo assim o custo do projeto.

Devido ao projeto de testes está definido e delimitado a partir do cronograma do projeto de desenvolvimento, é importante que seja dado foco aos pontos críticos do sistema, os quais são de grande importância para o negócio e que caso não sejam tratados adequadamente, podem gerar prejuízos inestimáveis quando o software estiver na fase de produção. A falta de controle ou excesso de preciosismo por parte da equipe de teste pode tornar os testes custosos para o projeto, podendo fazer a empresa considerar a interrupção dos testes.

Testar software não é somente executá-lo com a intenção de encontrar erros, existem várias atividades como: planejamento e controle, escolha das condições de teste, modelagem dos casos de teste, checagem dos resultados, avaliação de conclusão dos testes, geração de relatórios como também a revisão dos documentos.

2. Técnicas de Teste

Técnica é o processo que assegura o funcionamento correto de alguns aspectos do software ou de uma unidade do software. Existem muitas maneiras de se testar um software. Mesmo assim, existem as técnicas que sempre foram muito utilizadas em sistemas desenvolvidos sobre linguagens estruturadas que ainda hoje têm grande valia para os sistemas orientados a objeto. Apesar de os paradigmas de desenvolvimento ser completamente diferentes, o objetivo principal destas técnicas continua a ser o mesmo: encontrar falhas no software.
Existem dois tipos de técnicas de teste de software:

2.1. Caixa-Branca

Também chamada de teste estrutural, a técnica de caixa-branca, ocorrem durante a codificação do programa, avalia o comportamento interno do componente de software. Os aspectos avaliados nesta técnica de teste dependerão da complexidade e da tecnologia que determinarem a construção do componente de software. O testador tem acesso ao código fonte da aplicação e pode construir códigos para efetuar a ligação de bibliotecas e componentes. Este tipo de teste é desenvolvido analisando o código fonte e elaborando casos de teste que cubram todas as possibilidades do componente de software. Dessa maneira, todas as variações relevantes originadas por estruturas de condições são testadas.

Um exemplo bem prático desta técnica de teste é o uso da ferramenta livre JUnit para desenvolvimento de classes de teste para testar classes ou métodos

desenvolvidos em Java. Também se enquadram nessa técnica testes manuais ou testes efetuados com apoio de ferramentas para verificação de aderência a boas práticas de codificação reconhecidas pelo mercado de software. A aderência a padrões e boas práticas visa principalmente a diminuição da possibilidade de erros de codificação e a busca de utilização de comandos que gerem o melhor desempenho de execução possível. Apesar de muitos desenvolvedores alegarem que não há ganhos perceptíveis com essa técnica de teste aplicada sobre unidades de software, devemos lembrar que, no ambiente produtivo, cada programa pode vir a ser executado milhares ou milhões de vezes em intervalos de tempo pequenos. É na realidade de produção que a soma dos aparentes pequenos tempos de execução e consumo de memória de cada programa poderá levar o software a deixar de atender aos objetivos esperados. A técnica de teste de caixa-branca é recomendada para as fases de teste de unidade e teste de integração, cuja responsabilidade principal fica a cargo dos desenvolvedores do software, que por sua vez conhecem bem o código fonte produzido.

Esta técnica possui tipos de testes específicos que serão listados abaixo.

- **Teste de Execução:** Verifica os tempos de resposta, de processamento e o desempenho (performance), avaliando o comportamento do software no ambiente de produção e verificando se as premissas de desempenho são atendidas. Em um sistema que possui dez módulos diferentes e que foi desenvolvido por equipes diferentes, o teste de execução avalia o sistema como um todo, é como se o teste de execução fosse um “play” no sistema.
- **Teste de Estresse:** Avalia o comportamento do software sob condições críticas, tais como restrições significativas de memória, espaço em disco, etc., ou seja, coloca o software sob condições mínimas de operação.
- **Teste de Recuperação:** A recuperação é a capacidade de reiniciar operações após a perda da integridade de uma aplicação como, por exemplo: Ao desligar o computador, queda de energia elétrica, entre outros. O teste de recuperação garante a continuidade das operações após um desastre.
- **Teste de Operação:** Avalia o processo e sua execução, são desenhados para estabelecer se o sistema é executável durante a operação normal, é um tipo de teste muito específico, depende do software a ser testado um exemplo é o software de “Call Center”.
- **Teste de Conformidade:** Verifica se o software foi desenvolvido de acordo com padrões, normas, procedimentos e guias de TI.
- **Teste de Segurança:** Avalia a adequação dos procedimentos de proteção e as contramedidas projetadas, para garantir a confidencialidade das informações e a proteção dos dados contra o acesso não autorizado de terceiros.

2.2. Caixa-Preta

Também chamada de teste funcional, que garante o cumprimento dos requisitos pelo sistema, a técnica de caixa-preta avalia o comportamento externo do componente de software, sem se considerar o comportamento interno do mesmo. Dados de entrada são fornecidos, o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. Como detalhes de implementação não são considerados, os casos de teste são todos derivados da especificação.

Quanto mais entradas são fornecidas, mais rico será o teste. Numa situação ideal todas as entradas possíveis seriam testadas, mas na ampla maioria dos casos isso é impossível. Outro problema é que a especificação pode estar ambígua em relação ao sistema produzido, e como resultado as entradas especificadas podem não ser as mesmas aceitas para o teste. Uma abordagem mais realista para o teste de caixa-preta é escolher um subconjunto de entradas que maximize a riqueza do teste. Podem-se agrupar subconjuntos de entradas possíveis que são processadas similarmente, de forma que testar somente um elemento desse subconjunto serve para averiguar a qualidade de todo o subconjunto. Por exemplo, em um sistema que aceita um inteiro como entrada, testar todos os casos possíveis pode gerar pelo menos dezenas de milhares de casos de testes distintos. Entretanto, a partir da especificação do sistema, pode-se encontrar um subconjunto de inteiros que maximizem a qualidade do teste. Depende do propósito do sistema, mas casos possíveis incluem inteiros pares, inteiros ímpares, zero, inteiros positivos, inteiros negativos, o maior inteiro, o menor inteiro.

Essa técnica é aplicável a todas as fases de teste – teste unitário, teste de integração, teste de sistema e teste de aceitação. A aplicação de critérios de teste leva o testador a produzir um conjunto de casos de teste (ou situações de teste). A aplicação do critério de Particionamento de Equivalência (ou uso de classes de equivalência) permite avaliar se a quantidade de casos de teste produzida é coerente. O Particionamento de Equivalência se baseia em testar subconjuntos dos dados e não todos os dados possíveis - o que seria exaustivo e às vezes impossível -, pode-se assumir que as classes de equivalência serão tratadas da mesma maneira, pois um único elemento da classe se comporta como um representante dessa classe. A partir das classes de equivalência identificadas pode-se usar a Análise de Valor Limite, o testador construirá casos de teste que atuem nos limites superiores e inferiores destas classes, de forma que um número mínimo de casos de teste permita a maior cobertura de teste possível. Outro critério é o Grafo Causa-Efeito, que consiste em utilizar a ideia de grafos para transformar entradas de dados em causas e saídas de dados em efeitos. Esse grafo é posteriormente convertido para tabela de decisão e este para casos de teste. Por fim, tem-se o critério de Error-Guessing, que é uma técnica em que os analistas de teste, por meio da experiência e intuição, supõem tipos prováveis de erro.

Uma abordagem no desenvolvimento do teste de caixa-preta é o teste baseado na especificação, de forma que as funcionalidades são testadas de acordo com os requisitos. Apesar de necessário, esse tipo de teste é insuficiente para identificar certos riscos num projeto de software.

Esta técnica também possui tipos de testes específicos que serão listados abaixo.

- **Teste de Requisitos:** Verifica se o sistema é executado conforme o que foi especificado. São realizados através da criação de condições de testes e cheklists de funcionalidades.
- **Teste de Regressão:** Testa se algo mudou em relação ao que já estava funcionando corretamente, ou seja, é voltar a testar segmentos já testados após uma mudança em outra parte do software. Os testes de regressão devem ser feitos tanto no software quanto na documentação.
- **Teste de Tratamento de Erros:** Determina a capacidade do software de tratar transações incorretas. Esse tipo de teste requer que o testador pense negativamente e conduza testes como: entrar com dados cadastrais impróprios, tais como preços, salários, etc., para determinar o

comportamento do software na gestão desses erros. Produzir um conjunto de transações contendo erros e introduzi-los no sistema para determinar se este administra os problemas.

- **Teste de Suporte Manual:** Verifica se os procedimentos de suporte manual estão documentados e completos, determina se as responsabilidades pelo suporte manual foram estabelecidas.
- **Teste de Interconexão:** Garante que a interconexão entre os softwares de aplicação funcione corretamente. Pois, softwares de aplicação costumam estar conectados com outros softwares de mesmo tipo.
- **Teste de Controle:** Assegura que o processamento seja realizado conforme sua intenção. Entre os controles estão a validação de dados, a integridade dos arquivos, as trilhas de auditoria, o backup e a recuperação, a documentação, entre outros.
- **Teste Paralelo:** Comparar os resultados do sistema atual com a versão anterior determinando se os resultados do novo sistema são consistentes com o processamento do antigo sistema ou da antiga versão. O teste paralelo exige que os mesmos dados de entrada rodem em duas versões da mesma aplicação. Por exemplo: caso a versão mude e os requisitos não, os dados de saída das duas versões devem ser iguais.

3. Fases de Teste

Uma prática comum é testar o software após uma funcionalidade ser desenvolvida, e antes dela ser implantada no cliente, por um grupo de profissionais diferente da implementação. Essa prática pode resultar na fase de teste ser usada para compensar atrasos do projeto, comprometendo o tempo devotado ao teste. Outra prática é começar o teste no mesmo momento que o projeto, num processo contínuo até o fim do projeto.

3.1. Teste de Unidade

Também conhecida como Teste Unitário. É a fase do processo de teste em que se testam as menores unidades de software desenvolvidas (pequenas partes ou unidades do sistema). O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código. Assim, o objetivo é o de encontrar falhas de funcionamento dentro de uma pequena parte do sistema funcionando independentemente do todo.

3.2. Teste de Integração

Na fase de teste de integração o objetivo é encontrar falhas provenientes da integração interna dos componentes de um sistema. Geralmente os tipos de falhas encontradas são de envio e recebimento de dados. Por exemplo, um objeto A pode estar aguardando o retorno de um valor X ao executar um método do objeto B, porém este objeto B pode retornar um valor Y, desta forma gerando uma falha. Não faz parte do escopo dessa fase de teste o tratamento de interfaces com outros sistemas (integração entre sistemas). Essas interfaces são testadas na fase de teste de sistema, apesar de, a critério do gerente de projeto, estas interfaces podem ser testadas mesmo antes de o sistema estar plenamente construído.

3.3. Teste de Sistema

Na fase de Teste de Sistema o objetivo é executar o sistema sob ponto de vista de seu usuário final, varrendo as funcionalidades em busca de falhas. Os testes são executados em condições similares – de ambiente, interfaces sistêmicas e massas de dados – àquelas que um usuário utilizará no seu dia-a-dia de manipulação do sistema. De acordo com a política de uma organização podem ser utilizadas condições reais de ambiente, interfaces sistêmicas e massas de dados.

3.4. Teste de Aceitação

Fase de Teste em que o teste é conduzido por usuários finais do sistema. Os testes são realizados, geralmente, por um grupo restrito de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado. Teste formal conduzido para determinar se um sistema satisfaz ou não seus critérios de aceitação e para permitir ao cliente determinar se aceita ou não o sistema. Validação de um software pelo comprador, pelo usuário ou por terceira parte, com o uso de dados ou cenários especificados ou reais. Pode incluir testes funcionais, de configuração, de recuperação de falhas, de segurança e de desempenho.

3.5. Teste de Operação

Fase de Teste em que o teste é conduzido pelos administradores do ambiente final onde o sistema ou software entrará em ambiente produtivo. Vale ressaltar que essa fase é aplicável somente a sistemas de informação próprios de uma organização, cujo acesso pode ser feito interna e/ou externamente a essa organização. Nessa fase de teste devem ser feitas simulações para garantir que a entrada em produção do sistema será bem sucedida. Envolve testes de instalação, simulações com backup e restore das bases de dados, etc. Em alguns casos um sistema entrará em produção para substituir outro e é necessário garantir que o novo sistema continuará garantindo o suporte ao negócio.

3.6. Teste de Regressão

Teste de Regressão é uma fase de teste aplicável a uma nova versão de software ou à necessidade de se executar um novo ciclo de teste durante o processo de desenvolvimento. Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Inclui-se nesse contexto a observação de fases e técnicas de teste de acordo com o impacto de alterações provocado pela nova versão ou ciclo de teste. Para efeito de aumento de produtividade e de viabilidade dos testes, é recomendada a utilização de ferramentas de automação de testes, de forma que, sobre a nova versão ou ciclo de teste, todos os testes anteriores possam ser reexecutados com maior agilidade.

3.7. Testes Alpha, Beta e Gama.

Em casos especiais de processos de desenvolvimento de software – Sistemas Operacionais, Sistemas Gerenciadores de Bancos de Dados (SGBD), e outros softwares comerciais disponibilizados no mercado nacional e internacional – os testes requerem fases também especiais antes do produto ser disponibilizado a todos os usuários.

O período entre o término do desenvolvimento e a entrega é conhecido como fase alpha e os testes executados nesse período, como testes alpha.

PRESSMAN afirma que o teste alpha é conduzido pelo cliente no ambiente do desenvolvedor, com este “olhando sobre o ombro” do usuário e registrando erros e problemas de uso.

Completada a fase alpha de testes, são lançadas a grupos restritos de usuários, versões de teste do sistema denominadas versões beta. O Teste Beta também é um teste de aceitação voltado para softwares cuja distribuição atingirá grande número de usuários de uma ou várias empresas compradoras. PRESSMAN afirma que o teste beta é conduzido em uma ou mais instalações do cliente, pelo usuário final do software. Diferente do teste alpha, o desenvolvedor geralmente não está presente. Consequentemente, o teste beta é uma aplicação “ao vivo” do software num ambiente que não pode ser controlado pelo desenvolvedor. O cliente registra todos os problemas (reais ou imaginários) que são encontrados durante o teste beta e os relata ao desenvolvedor em intervalos regulares. Com o resultado dos problemas relatados durante o teste beta, os engenheiros de software fazem modificações e depois se preparam para liberar o produto de software para toda a base de clientes.

Os testes Gama não são propriamente testes de software. A comunidade do teste de software usa este termo de forma sarcástica referindo-se aos produtos que são mal testados e são entregues aos usuários (“end-users”) para que estes encontrem os defeitos já em fase de produção.

4. Ciclo de Vida do Teste

O Ciclo de Vida dos Testes é composto pelas 5 fases a seguir.

- **Planejamento:** Nesta fase é elaborada a Estratégia de Teste e o Plano de Teste.
- **Preparação:** O objetivo desta fase é preparar o Ambiente de Teste (equipamentos, pessoal, ferramentas de automação, massa de testes) para que os testes sejam executados conforme planejados.
- **Especificação:** Nesta fase temos as seguintes atividades: Elaborar/ Revisar casos de testes e Elaborar/ Revisar roteiros de testes.
- **Execução:** Os testes são executados e os resultados obtidos são registrados.
- **Entrega:** Esta é a última fase do ciclo de vida de testes, onde o projeto é finalizado e toda documentação é finalizada e arquivada.

5. Papeis e Responsabilidades

Os principais participantes no processo de testes são:

- **Gerente de Teste:** Tem como papel defender a qualidade dos testes, planejar e gerenciar os recursos e resolver os problemas que representam obstáculos ao esforço de teste.
- **Líder de Teste:** Pessoa responsável pela liderança de um projeto de teste específico, normalmente relacionado a um projeto de desenvolvimento, seja um projeto novo ou uma manutenção.
- **Analista de Teste:** Elabora e modela os casos e roteiros de testes. Deve focar seu trabalho nas técnicas de teste adequadas à fase de teste trabalhada.
- **Arquiteto de Teste:** É responsável por montar a infra-estrutura de testes como: ambiente, ferramentas, capacitação da equipe, entre outros.

- **Testador:** Executa os testes, o mesmo deve observar as condições de teste e respectivos passos de teste documentados pelo analista de teste e evidenciar os resultados de execução.
- **Automatizador:** Tem como papel automatizar as situações de teste em ferramentas observando as condições de teste documentadas pelo analista de teste e automatizar a execução desses testes na ferramenta utilizada. Normalmente são gerados scripts de teste que permitam a execução de ciclos de teste sempre que se julgar necessário.

Uma pessoa pode assumir mais de um dos papéis citados acima como, por exemplo, um testador pode exercer o papel de um automatizador de testes também.

6. Conclusão

O teste de software é uma das atividades mais custosas do processo de desenvolvimento de software, pois pode envolver uma quantidade significativa dos recursos de um projeto. Diferentes categorias de aplicações requerem uma preocupação diferenciada com as atividades de teste. Um ponto bastante importante para a viabilização da aplicação de teste de software é a utilização de uma infraestrutura adequada. Realizar testes não consiste simplesmente na geração e execução de casos de teste, mas envolvem também questões de planejamento, gerenciamento e análise de resultados. Após ter sua estratégia de teste definida, tente buscar por ferramentas que se encaixem na sua estratégia. Isso pode reduzir significantemente o esforço de tal tarefa. Além disso, é importante ressaltar que diferentes tipos de aplicações possuem diferentes técnicas de teste a serem aplicadas. Cada tipo de aplicação possui características especificas que devem ser consideradas no momento da realização dos testes. O conjunto de técnicas apresentado neste artigo cobre diversas características comuns a muitas categorias de software, mas não é completo.

Para finalizar, conclui-se que testes bem planejados e automatizados permitem execuções em grande escala, com rapidez e precisão, detectando as não conformidades do software, iniciando as correções pela equipe de desenvolvimento o mais breve possível. O que dá ao software mais confiabilidade além de reduzir o custo do projeto.

Referências

[1] BSTQB – Brazilian Software Testing Qualifications Board. http://www.bstqb.org.br/uploads/docs/syllabus_2007br.pdf.

[2] Rios, Emerson; Cristalli, Ricardo; Moreira, Trayahú & Bastos, Aderson. Base de Conhecimento em Teste de Software. 2ª Edição – S. Paulo, Martins Fontes, 2007.

[3] MYERS, Glenford J. The Art of Software Testing 2 ed. Nova Jérsei: John Wiley & Sons.

[4] PRESSMAN, R. S. *Engenharia de Software*. [S.l.]: McGraw Hill.

[5] Wikipédia. “Teste de Software”. http://pt.wikipedia.org/wiki/Teste_de_software.

[6] Gustavo Quezada. “Papéis e Responsabilidades na Equipe de Testes”. http://www.alats.org.br/Default.aspx?tabid=206&mid=1073&ctl=Details&ItemID=109.