Teste de software

Origem: Wikipédia, a enciclopédia livre.

O **teste do software** é a investigação do software a fim de fornecer informações sobre sua qualidade em relação ao contexto em que ele deve operar, se relaciona com o conceito de verificação e validação. Isso inclui o processo de utilizar o produto para encontrar seus defeitos.

O teste é um processo realizado pelo testador de software, que permeia outros processos da engenharia de software, e que envolve ações que vão do levantamento de requisitos até a execução do teste propriamente dito.

Índice

Visão geral

Não se pode garantir que todo software funcione corretamente, sem a presença de erros,[1] visto que os mesmos muitas vezes possuem um grande número de estados com fórmulas, atividades e algoritmos complexos. O tamanho do projeto a ser desenvolvido e a quantidade de pessoas envolvidas no processo aumentam ainda mais a complexidade. Idealmente, toda

permutação possível do software deveria ser testada. Entretanto, isso se torna impossível para a ampla maioria dos casos devido à quantidade impraticável de possibilidades. A qualidade do teste acaba se relacionando à qualidade dos profissionais envolvidos em filtrar as permutações relevantes.[2]

Falhas podem ser originadas por diversos motivos. Por exemplo, a especificação pode estar errada ou incompleta, ou pode conter requisitos impossíveis de serem implementados, devido a limitações de hardware ou software. A implementação também pode estar errada ou incompleta, como um erro de um algoritmo. Portanto, uma falha é o resultado de um ou mais defeitos em algum aspecto do sistema.

O teste de software pode ser visto como uma parcela do processo de qualidade de software. A qualidade da aplicação pode e, normalmente, varia significativamente de sistema para sistema.

Os atributos qualitativos previstos na norma ISO 9126 são:

- Funcionalidade
- Confiabilidade
- Usabilidade
- Eficiência
- Manutenibilidade
- Portabilidade

De forma geral, mensurar o bom funcionamento de um software envolve compará-lo com elementos como especificações, outros softwares da mesma linha, versões anteriores do mesmo produto, inferências pessoais, expectativas do cliente, normas relevantes, leis aplicáveis, entre outros. Enquanto a especificação do software diz respeito ao processo de verificação do software, a expectativa do cliente diz respeito ao processo de validação do software. Por meio da verificação será analisado se o produto foi feito corretamente, se ele está de acordo com os requisitos especificados. Por meio da validação será analisado se foi feito o produto correto, se ele está de acordo com as necessidades e expectativas do cliente.

Um desenvolvimento de software organizado tem como premissa uma metodologia de trabalho. Esta deve ter como base conceitos que visem a construção de um produto de software de forma eficaz. Dentro desta metodologia estão definidos os passos necessários para chegar ao produto final esperado.

Assim, quando se segue uma metodologia para o desenvolvimento de um produto de software, espera-se um produto final que melhor agrade tanto aos clientes quanto ao próprio fornecedor, ou seja, a empresa de desenvolvimento. Observando este aspecto, não faz sentido iniciar a construção de um produto de software sem ter uma metodologia de trabalho bem solidificada e que seja do conhecimento de todos os envolvidos no processo. Porém, além de uma crescente demanda por softwares de qualidade, as empresas de desenvolvimento de software sofrem cada vez mais pressão por parte dos clientes para que o produto seja entregue num curto período de tempo. Este fato pode fazer com que uma sólida metodologia de trabalho acabe por se desequilibrar.

Independentemente da metodologia de trabalho empregada no desenvolvimento de um software, para que se obtenha um produto final com um certo nível de qualidade é imprescindível a melhoria dos processos de engenharia de software.

Uma maneira viável para se assegurar a melhoria de tais processos seria tomar como base modelos sugeridos por entidades internacionais respeitadas no assunto. Dentro de uma gama de modelos, sejam eles para situações e ambientes específicos ou para soluções genéricas, existem alguns que são mais utilizados e tidos como eficientes, como por exemplo os SW-CMM, SE-CMM, ISO/IEC 15504 e o mais conhecido CMMI.

Outro fator com grande influência sobre a qualidade do software a ser produzido é o que diz respeito aos testes que serão executados sobre tal produto. Todas as metodologias de desenvolvimento de software têm uma disciplina dedicada aos testes. Atualmente esta é uma tarefa indispensável, porém muitas vezes efetuada de maneira ineficiente, seja pelo subestimar dos que desenvolvem, pela falta de tempo ou mesmo pela falta de recursos humanos e financeiros.

De acordo com um estudo conduzido pelo NIST em 2002, os defeitos resultam num custo anual de 59,5 bilhões de dólares à economia dos Estados Unidos. Mais de um terço do custo poderia ser evitado com melhorias na infraestrutura do teste de software.[3]

Princípios

Para Myers (2004), há princípios vitais para o teste de software. O caso de teste deve definir a saída esperada, de forma a reduzir a interpretação do critério de sucesso. A saída da execução do teste deve ser exaustivamente analisada. Os casos de teste devem verificar não somente as condições inválidas de execução, como também as condições válidas. Outro conceito apresentado é utilizar pessoas e organizações diferentes para a implementação e para a verificação. A entidade de teste possui uma visão destrutiva do sistema, em busca de erros, enquanto a entidade de programação possui uma visão construtiva, em busca da implementação de uma especificação.

Myers também aborda o esforço para se construir casos de teste. Deve-se evitar testes descartáveis, pois a qualidade do teste piora gradualmente com as iterações de desenvolvimento. Em contrapartida, há o teste de regressão, que permite quantificar a evolução da qualidade de software, mantendo e executando novamente testes realizados anteriormente.

O mesmo autor afirma que, diferente do que se poderia considerar senso comum, a probabilidade de existência de erros num certo trecho de código é proporcional à quantidade de erros já encontrada anteriormente. Basicamente, erros aparecem em grupos. Trechos específicos de código de um software qualquer estão mais propensos a ter erros que outros.

Técnicas

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 serem completamente diferentes, o objetivo principal destas técnicas continua a ser o mesmo, encontrar falhas no software. Abaixo estão descritas algumas das técnicas mais conhecidas.

Caixa-branca

Também chamada de teste estrutural ou orientado à lógica, a técnica de caixa-branca avalia o comportamento interno do componente de software. Essa técnica trabalha diretamente sobre o código fonte do componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos, teste de caminhos lógicos, códigos nunca executados.

Os aspectos avaliados nesta técnica de teste dependerão da complexidade e da tecnologia que determinarem a construção do componente de software, cabendo portanto avaliação de mais aspectos que os citados anteriormente. 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.

Caixa-preta

Também chamada de teste funcional, teste comportamental, orientado a dado ou orientado a entrada e saída, a técnica de caixa-preta avalia o comportamento externo do componente de software, sem se considerar o comportamento interno do mesmo.[4] 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.[5] 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.[6] Uma abordagem mais realista para o teste de caixa-preta é escolher um subconjunto de entradas que maximize a riqueza do teste. Pode-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.[7]

Caixa-cinza

A técnica de teste de caixa-cinza é uma mescla do uso das técnicas de caixa-preta e de caixa-branca. Isso envolve ter acesso a estruturas de dados e algoritmos do componente a fim de desenvolver os casos de teste, que são executados como na técnica da caixa-preta. Manipular entradas de dados e formatar a saída não é considerado caixa-cinza pois a entrada e a saída estão claramente fora da caixa-preta. A caixa-cinza pode incluir também o uso de engenharia reversa para determinar por exemplo os limites superiores e inferiores das classes, além de mensagens de erro.

Regressão

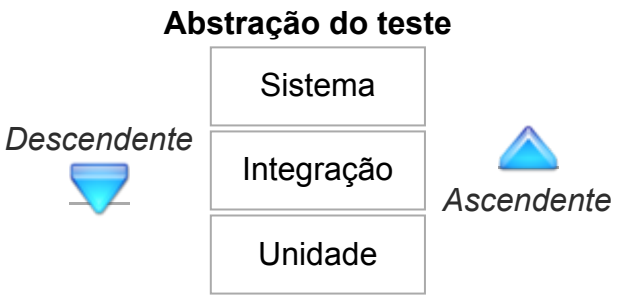
Essa é uma técnica de teste aplicável a uma nova versão de software ou à necessidade de se executar um novo ciclo de teste durante o processo de desenvolvimento. Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste 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 teste, de forma que, sobre a nova versão ou ciclo de teste, todos os testes anteriores possam ser executados novamente com maior agilidade.

Técnicas não funcionais

São técnicas utilizadas para verificar a operação correta do sistema em relação a casos inválidos ou inesperados de entrada. Outras técnicas de teste existem para testar aspectos não-funcionais do software, como por exemplo, a adequação a restrições de negócio, adequação a normas, ou restrições tecnológicas. Em contraste às técnicas funcionais mencionadas acima, que verificam a produção pelo sistema de respostas adequadas de suas operações, de acordo com uma especificação, as técnicas não funcionais verificam atributos de um componente ou sistema que não se relacionam com a funcionalidade (por exemplo, confiabilidade, eficiência, usabilidade, manutenibilidade e portabilidade)[8].

Uma delas é o uso conjunto de teste de desempenho e teste de carga, que verifica se o software consegue processar grandes quantidades de dados, e nas especificações de tempo de processamento exigidas, o que determina a escalabilidade do software. O teste de usabilidade é necessário para verificar se a interface de usuário é fácil de se aprender e utilizar. Entre verificações cabíveis estão a relação da interface com conhecimento do usuário, a compreensibilidade das mensagens de erro e a integridade visual entre diferentes componentes.[9] Já o teste de confiabilidade é usado para verificar se o software é seguro em assegurar o sigilo dos dados armazenados e processados. O teste de recuperação é usado para verificar a robustez do software em retornar a um estado estável de execução após estar em um estado de falha.

Fases ou Níveis



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.

Em contrapartida, algumas práticas emergentes como a programação extrema e o desenvolvimento ágil focam o modelo de desenvolvimento orientado ao teste. Nesse processo, os testes de unidade são escritos primeiro (TDD), por engenheiros de software. Antes da implementação da unidade em questão, o teste falha. Então o código é escrito, passando incrementalmente em porções maiores dos casos de teste. Os testes são mantidos junto com o resto do código fonte do software, e geralmente também integra o processo de construção do software. Existe também o Behavior Driven Development (BDD) ou Desenvolvimento Guiado por Comportamento.

Teste de unidade

Também conhecida como teste unitário ou teste de módulo, é a fase em que se testam as menores unidades de software desenvolvidas (pequenas partes ou unidades do sistema).[10] O universo alvo desse tipo de teste são as subrotinas, métodos, classes 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.

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 transmissão de dados. Por exemplo, um componente A pode estar aguardando o retorno de um valor X ao executar um método do componente B; porém, B pode retornar um valor Y, gerando uma falha. O teste de integração conduz ao descobrimento de possíveis falhas associadas à **interface do sistema**. 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.

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 em relação aos objetivos originais. 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.

Teste de aceitação

Geralmente, os testes de aceitação são realizados por um grupo restrito de usuários finais do sistema, que 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.

Teste de operação

Nessa fase o teste é conduzido pelos administradores do ambiente final em que 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 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 cópia de segurança dos bancos 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.

Teste de regressão

O teste de regressão é um teste que geralmente se aplica a uma versão mais recente de uma aplicação ou para executar um novo ciclo de testes durante o desenvolvimento para averiguar se todas as funções estão sendo executadas sem nenhum erro. Esse teste tem a função de aplicar a uma nova versão de uma aplicação ou a um ciclo de uma aplicação, todos os testes que foram feitos anteriormente. Nos testes feitos tem modificações para se adequarem as mudanças que houve de uma versão para a outra de uma aplicação para que os testes sejam viáveis a versão mais atualizada.

Testes alpha, beta e gama

Alpha

O teste apha são os testes que são feitos entre o término do desenvolvimento de uma aplicação e a sua entrega, e é feito com a observação do desenvolvedor, onde os usuários vão registrando os erros e problemas de uso da aplicação.

Beta

Após o termino dos testes alpha, são abertos os testes beta, onde um grupo restrito de usuários podem testar a aplicação. Geralmente o teste beta atinge um grande número de usuários, ao contrario do teste alpha esse teste geralmente é feito sem a presença do desenvolvedor. Ou seja, o teste beta é feito em um ambiente sob o qual o desenvolvedor não tem controle. Os usuários reportam os problemas encontrados na aplicação durante os testes. Com o resultado dos problemas reportados pelo usuário, os engenheiros de software fazem as devidas correções para que a aplicação possa ser disponibilizada para o mercado onde o público geral tenha acesso.

Gama

O gama basicamente lança produtos que foram "mal testados" ao mercado e os usuários já tem acesso a aplicação, mesmo que com muitos erros, e os problemas que os usuários encontram são corrigidos quando a aplicação já esta disponível para os usuários finais.

O Ciclo de Vida dos Testes

O Ciclo de Vida dos Testes é composto de 5 fases: Planejamento, Preparação, Especificação, Execução e Entrega.

Planejamento

Nesta fase é elaborada a Estratégia de Teste e o Plano de Teste, em cima dos requisitos.

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.

Artefatos

O processo de teste de software pode produzir diversos artefatos. O caso de teste geralmente consiste de uma referência a um identificador ou requisito de uma especificação, pré-condições, eventos, uma série de passos a se seguir, uma entrada de dados, uma saída de dados, resultado esperado e resultado obtido. A série de passos (ou parte dela) pode estar contida num procedimento separado, para que possa ser compartilhada por mais de um caso de teste. O roteiro de teste (https://webinsider.com.br/um-roteiro-para-planejar-testes-de-software/)

Um *script* de teste é a combinação de um caso de teste, um procedimento de teste e os dados do teste. Uma coleção de casos de teste é chamada de suite de teste. Geralmente, ela também contém instruções detalhadas ou objetivos para cada coleção de casos de teste, além de uma seção para descrição da configuração do sistema usado.

A especificação de teste é chamada plano de teste.

Referências

1. MYERS, 2004, p. 8
2. MYERS, 2004, p. 5
3. Michael Newman (28 de junho de 2002). «Software Errors Cost U.S. Economy $59.5 Billion Annually: NIST Assesses Technical Needs of Industry to Improve Software-Testing» (http://www.nist.gov/public_affairs/releases/n02-10.htm) (em inglês). NIST. Consultado em 17 de novembro de 2008
4. MYERS, 2004, p. 9
5. MYERS, 2004, p. 10
6. Jiantao Pan (1999). «Software Testing» (http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/) (em inglês). Universidade Carnegie Mellon. Consultado em 1 de dezembro de 2008
7. James Bach (Junho de 1999). «Risk and Requirements-Based Testing» (http://www.satisfice.com/articles/requirements_based_testing.pdf) (PDF) (em inglês). IEEE. Consultado em 17 de novembro de 2008
8. [1] (http://www.bstqb.org.br/?q=node/570)
9. MYERS, 2004, p. 136
10. MYERS, 2004, p. 91

Bibliografia

- KOSCIANSKI, A., Soares (2006). *M. S. Qualidade de Software*. [S.l.]: Novatec
- PRESSMAN, R. S. (2002). *Engenharia de Software*. [S.l.]: McGraw Hill
- MYERS, Glenford J. (2004). *The Art of Software Testing* 2 ed. Nova Jérsei: John Wiley & Sons. ISBN 0-471-46912-2
- ARAYA, MARCIO SAN (*Teacher FAC SENAC*)

Ver também

- Anexo:Lista de instituições pela qualidade
- Qualidade de software
- Gestão da qualidade
- Verificação formal
- Otimização em engenharia de software

Ligações externas

- Guia de gerenciamento de teste (http://www.ruleworks.co.uk/testguide)
- Comunidade Testes de Software (http://www.testesdesoftware.blogspot.com)
- Empresas testes de software (http://www.testcompanies.com)