CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS V – DIVINÓPOLIS - TÉCNICO EM INFORMÁTICA – 3º B
DISCIPLINA PROJETOS DE SISTEMA

Testes de Software

André Almeida Gonçalves

Júlia Elias Morato

Nathália Cristian Ferreira de Oliveira

Professor: Michel Pires da Silva

Divinópolis, 17 de agosto, 2015.

SUMÁRIO

1. INTRODUÇÃO

Hoje em dia percebemos que os softwares estão cada vez mais presentes em nossa vida diária, por meio de diversos dispositivos, tais como celulares, computadores e *tablets*. Como consequência disso, vemos que o mercado de softwares está cada vez mais competitivo, exigindo aplicativos mais estáveis, complexos e livres de erros. Muitas vezes, ao utilizar programas que tenham sido desenvolvidos sem a devida precaução e detecção antecipada de erros, os usuários são submetidos à muitos problemas e prejuízos causados por aplicativos que deveriam facilitar e melhorar sua vida.

Para resolver essa situação de má funcionamento de softwares, é necessário que os desenvolvedores utilizem testes para reduzir o risco de ocorrência de defeitos que possam prejudicar os clientes. Além disso, quanto mais cedo forem encontrados os defeitos no software, menor é o custo para resolver o problema, já que depois que o programa está em funcionamento, é muito trabalhoso alterar seu funcionamento, que inclusive pode já ter causado algum dano ao usuário.

Para se conduzir e avaliar a qualidade da atividade de teste têm-se as **técnicas de teste** funcional e estrutural. Tais técnicas diferenciam-se pela origem da informação utilizada na avaliação e construção dos conjuntos de casos de teste.

A **Técnica Funcional** também é conhecida como teste caixa preta pelo fato de tratar o software como uma caixa em que o conteúdo é desconhecido e da qual só é possível visualizar o lado externo, ou seja, os dados de entrada fornecidos e as respostas produzidas como saída.

Nos testes de **Técnica Estrutural**, também conhecida como teste caixa branca, os aspectos de implementação são fundamentais na escolha dos casos de teste. O teste estrutural baseia-se no conhecimento da estrutura interna da implementação. Essa técnica é vista como complementar à técnica funcional e informações obtidas pela aplicação desses critérios têm sido consideradas relevantes para as atividades de manutenção, depuração e confiabilidade de software.

Em relação aos **tipos de teste**, verifica-se o **Teste de Regressão**, em que se 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. Aplicado na fase de manutenção do sistema, após sua implantação.

Por outro lado, temos o **Teste de Requisitos** que verifica se o sistema é executado conforme o que foi especificado. São realizados através da criação de condições de testes e cheklists de funcionalidades. Na etapa de requisitos do sistema é preciso definir requisitos testáveis, ou seja, os requisitos devem ser escritos com o intuito de ser testados futuramente para verificar se as especificações do sistema foram atendidas como o solicitado. Portanto, o teste de requisitos é o conjunto de teste elaborado para cada requisito registrado na fase de levantamento, essa técnica é essencial para verificar requisitos vagos ou ausentes. Além disso, o teste funciona como uma validação, isto é, ele demonstra que o sistema tem seus requisitos adequadamente implementados.

Existem também os **Testes de Unidade**, também conhecido como testes unitários. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. 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 .

Já o **Teste de Integração**, como o próprio nome já diz, visa encontrar falhas provenientes da integração dos componentes do sistema. Geralmente os tipos de falhas encontradas são de envio e recebimento de dados. Por exemplo, um objeto A pode estar esperando o retorno de um valor x ao executar um método do objeto B , porém este objeto B pode retornar um valor y , ou seja, diferente do esperado.

Considerando o **Teste de Controle**, vemos que sua função é assegurar que o processamento seja realizado conforme a intenção do idealizador. 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.

Utiliza-se também o **Teste de Execução**, que verifica os tempos de resposta, de processamento e o desempenho, 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.

Outro tipo de teste seria o **Teste de Estresse**, que 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.

Para resolver um problema recorrente foi criado o **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.

É preciso considerar também o **Teste de Operação**, que avalia o processo e sua execução, é 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”.

Por último, temos o **Teste de Segurança**, que 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.

O grupo escolheu o **Teste de Tratamento de Erros** para aprofundar seus estudos e o utilizar para verificar o funcionamento de seu software desenvolvido para o Trabalho de Conclusão de Curso do CEFET-MG de 2015 do curso de informática, **"Vacina.com"**. Maiores detalhes sobre esse teste são disponibilizados na seção 2.

2.TESTE DE TRATAMENTO DE ERROS

Em um artigo sobre os programadores da NASA de 1996 alegava-se que um dos softwares para controlar o ônibus espacial teve em suas onze últimas versões apenas 17 erros. Em outro artigo, este de 2003, alega-se que uma falha na aterrissagem da nave espacial Soyuz fora causada por falha de software.

Como toda atividade humana, erros que ocorrem com software trazem algum impacto que pode ser alto ou baixo dependendo da área em que o programa esteja atuando. Muito se tem pesquisado e horas intermináveis são gastas corrigindo problemas de software e procurando-se projetar os sistemas para que simplesmente não falhem, mas é muito difícil e em alguns casos impossível escrever código 100% à prova de falhas.

Quer seja por fatores humanos – alguém sempre descobre um jeito “não testado e não documentado” de utilizar o seu programa – ou por aspectos de falha de hardware, rede etc., os programas virão a falhar em algum momento. Cabe para aqueles que elaboram o software a tarefa de fazer um tratamento adequado dos erros para que estes não coloquem todos os dados a perder e causem eventuais prejuízos.

O Microsoft .NET Framework implementou um meio de capturar erros dos programas para que todo o código suscetível a eles rode dentro de um bloco protegido conhecido como “try...catch”. Este já estava presente também em outras linguagens orientadas a objeto, como C++ e Java.

O bloco “try...catch” representou uma grande evolução das linguagens de programação da Microsoft quanto ao tratamento de erros. É bem semelhante ao tratamento que existia antes em linguagens como o C++ e o Java e isto fez com que as linguagens do .NET Framework como C# e VB.NET não fiquem devendo nada a estas outras quanto ao tratamento de erro.

Devido à gravidade das consequências geradas pro erros em softwares, vê-se necessário implementar testes que verifiquem a aparição de tais erros em nosso sistema, visto que já utilizamos o sistema Try/Catch. É indispensável também a aparição de **fluxos de exceção**, que tratem erros que ocorram por má utilização do sistema pelo usuário ou tentativas de acesso inapropriado à partes reservadas do software ou banco de dados.

Por isso decidimos implementar o **Teste de Tratamento de Erros**, que 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.

Considerando a trajetória da criação dos computadores e implementações de software, percebe-se que sempre existiu a necessidade de realizar testes que previnam erros no sistema, e esses testes são utilizados até hoje em praticamente todos os softwares que podem ser considerados confiáveis, demonstrando a grande importância desse teste para a sociedade atual.

3. CONCLUSÃO

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, ou seja, atender aos requisitos impostos pelo mesmo. Realizar testes dentro de um processo com metodologia própria é o grande "X" da questão. 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.

REFERÊNCIAS

- http://www.linhadecodigo.com.br/artigo/2775/introducao-ao-teste-de-software.aspx - Acesso 14/09/2015
- http://pbjug-grupo-de-usuarios-java-da-paraiba.1393240.n2.nabble.com/attachment/2545944/0/Introducao%20Ao%20Teste%20De%20Software%20-%20Maldonado,Jose.pdf - Acesso 14/09/2015
- http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_ND_65.pdf - Acesso 14/09/2015
- http://www.devmedia.com.br/tratamento-adequado-de-erros-net-magazine-81/18963#ixzz3ljlqU4gh - Acesso 14/09/2015