COMO O TESTE DE SOFTWARE PODE AJUDAR O DESENVOLVIMENTO

Ricardo Zitelli de OLIVEIRA[1]
Álvaro Ferraz DARCE[2]

RESUMO: Um *software* não pode ser entregue ao cliente sem antes ser alcançada a certeza de que seus requisitos serão atendidos. Assim sendo, surge a necessidade de realização de testes de software para que eles possam ser distribuídos com um nível de qualidade aceitável pelo mercado. Dos testes passíveis de realização, surgem como exemplos os testes de componentes, os testes de integração, os testes de caixa-preta e os testes de caixa-branca.

Palavras-chave: Teste. Testes Componentes. Finalidade dos Testes. Caixa Preta e Branca.

1 INTRODUÇÃO

O presente artigo tem por objetivo dissertar sobre como o teste de software pode ajudar no desenvolvimento de um sistema e porque é tão importante sua aplicação, não somente quando o sistema estiver finalizado, mas também em durante o processo de desenvolvimento. Segundo Sammy (s.d., s.p.):

> O maior desafio das chamadas Fábricas de Software é entregar um produto funcional e com qualidade, visto que muitas delas se preocupam em analisar, requisitar e construir, deixando de lado as atividades referentes à avaliação da qualidade. Muitas vezes o orçamento pequeno, prazo apertado e cobranças acabam fazendo com que alguns processos tomem o lugar de outros que, a princípio, eram considerados menos importantes.

Há ainda o fato de que, mesmo testando o software, analisando linhas de códigos, realizando testes, muito provavelmente serão encontrados defeitos.

Por isso é necessário que os requisitos de sistema sejam bem definidos e modelados, pois, por meio deles, o analista de testes poderá elaborar um bom projeto de testes do *software*. Segundo Silva Filho (s.d.;s.p.):

---

[1] Discente do 4º ano do curso de Sistemas de Informação das Faculdades Integradas “Antonio Eufrásio de Toledo” de Presidente Prudente. ricardo_zitelli@unitoledo.br
[2] Docente do curso de Sistemas de Informação das Faculdades Integradas “Antonio Eufrásio de Toledo” de Presidente Prudente. Mestre em Ciência da Computação pela Universidade Estadual Paulista "Júlio de Mesquita Filho" alvaro@darce.com.br Orientador do trabalho.

> Desenvolver um sistema de software requer um processo, o qual informa um conjunto de atividades a serem realizadas, quem as executam, quais artefatos de entrada são necessários e quais artefatos de saída são produzidos. Nesse sentido, detectar erros ou quaisquer outros problemas como, por exemplo, inconsistência e falta de clareza é de suma importância de modo a tornar o processo mais efetivo sob o ponto de vista de custo.

Se um teste for aplicado de maneira incorreta ou se algum processo do sistema não for testado, o responsável pelo desenvolvimento do *software* (e também o cliente) poderá ter prejuízos. Um dos possíveis problemas seria a refatoração das porções de código não testadas, o que poderia acarretar uma grande demanda de tempo.

Na pior das hipóteses, o responsável poderá ser malvisto por seu cliente, que poderá expressar seu descontentamento com amigos do ramo, acarretando em má fama do profissional na região.

Para o desenvolvimento do mister elencado, o presente ensaio foi estruturado em alguns pontos principais. De proêmio, buscou-se analisar o que é o teste de software, para em seguida discorrer acerca de suas utilidades e quais são os métodos existentes para se proceder ao teste do sistema. Finalmente, foi enfrentada uma discussão acerca da possibilidade de se observar uma linha única de testes para *softwares*.

Quanto à metodologia utilizada, esta consistiu basicamente em pesquisa científica realizada em artigos e material impresso e digital acerca do assunto, levando-se em conta ainda, para efeitos de estruturação do presente, o método indutivo.

2 OS TESTES DE *SOFTWARE* NA INFORMÁTICA CONTEMPORÂNEA

Antes de passar-se a uma análise mais aprofundada a respeito de como a importância da realização de testes nos *softwares* progrediu juntamente com o avanço da informática, é necessário que eles sejam devidamente conceituados. Neste contexto, uma definição bastante completa é trazida por Pontes (s.d, s.p.):

> Testar um software e relatar impressões e não conformidades é fornecer um diagnóstico do estado da aplicação, e é muito importante que este diagnóstico seja o mais completo e preciso possível, porque provavelmente ele vai servir de base para tomadas de decisões em relação ao projeto que está sendo analisado. Testar uma aplicação é questioná-la, através de casos de teste e principalmente de observações, para analisar as respostas obtidas, pois estas podem revelar defeitos.

Do exposto, pode-se concluir que testar um sistema é proceder a uma análise pormenorizada das reações verificadas quando de sua execução prática. Basicamente, é executar o *software* para verificar que se ele apresenta alguma falha ou imperfeição que comprometa a sua eficácia.

De fato, assevera Meriat (s.d, s.p.) que "é um processo para determinar a qualidade do software desenvolvido".

Finalmente, Dias Neto (s.d, s.p.) conclui que "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".

Em sendo assim, pode-se afirmar que o teste consiste na busca por defeitos, falhas, imperfeições ou erros em um *software*.

Defeito, segundo Silva (s.d, s.p), "é qualquer imperfeição ou inconsistência no produto do software ou em seu processo", o que se justifica porque um sistema possui vários algoritmos complexos e inúmeros caminhos e brechas que podem passar despercebidos pelo desenvolvedor".

Se um defeito for encontrado, o responsável pelo teste deverá avisar o desenvolvedor para que ele possa realizar as devidas correções e aperfeiçoar seu sistema.

Nos primórdios da era tecnológica, onde as ciências da computação ainda estavam em seu nascedouro, a realização de testes nos sistemas desenvolvidos era vista como algo totalmente desnecessário e desprovido de utilidade e significância tal que merecessem um cuidado especial por parte do profissional analista de sistemas. Eis o que ensina Macoratti (s.d, s.p.)

> A tarefa de efetuar testes em software foi considerada secundária por muito tempo. Geralmente era vista como um castigo para o programador, ou como uma tarefa onde não se deveria gastar muito com tempo e investimentos. O tema esteve relegado a segundo plano, e, até alguns anos atrás não se encontrava muita literatura sobre o assunto.

A relutância em se conceder o adequado valor à realização dos testes de sistemas, em parte se justifica por aquilo que é conhecido como mito do teste. A respeito dele, leciona Tomaz (s.d., s.p.) que:

> Impedir que um sistema perca valor ao chegar às mãos do usuário com falhas cruciais ao seu funcionamento não é um trabalho limitado a apenas custos. O bom testador precisa conhecer o software como ninguém e ter alguém demonstrando esse conhecimento e assegurando maior qualidade no produto final é algo que acrescenta valor.

De fato, é possível perceber que o teste do sistema é um trabalho que não terá seus efeitos adstritos ao ramo da informática, já que um bom *software* consegue obter um lugar de destaque no mercado. E para que isto seja efetuado de maneira satisfatória, é necessário que o profissional que o realiza tenha um conhecimento um tanto quanto mais aprofundado dos fundamentos do sistema, com vistas a uma interpretação mais fiel dos resultados que o teste apresentar.

Outros fatores que, durante muito tempo fizeram com que os testes de sistemas ficassem numa posição secundária são os elevados custos financeiros que eles implicam. Reuters (2005, s.p.) menciona que a respeito deste assunto que:

> Estima-se que defeitos de software tenham custado cerca de US$ 60 bilhões no ano passado apenas nos Estados Unidos, de acordo com o National Institute of Standards and Technology. Empresas como corretoras de ações online e fabricantes de celulares estão dispostas a gastar muito dinheiro para evitá-los.

Os dados referem-se ao ano de 2004. Paralelamente a isto, não se pode perder de vista que há um elevado gasto de tempo com a correção dos vícios dos sistemas, justamente porque a descoberta da raiz dos problemas acusados pelos testes demanda análises aprofundadas de cada um dos elementos sobre os quais todo o programa desenvolvido se funda.

Entretanto, como alhures elencado, a partir de quando as premissas de mercado se tornaram mais presentes no quotidiano da informática, passando os profissionais da programação a entender que somente quando se busca a perfeição de um sistema ele se torna competitivo no mercado, que os testes passaram a ser tidos como vitais, abandonando o ostracismo em que outrora figuravam para alcançar lugar destaque no desenvolvimento da programação.

Assim, a se antes os testes eram vistos como algo sem importância, gradativamente passaram a ser vistos como uma ferramenta de eficácia única na busca pela excelência do desenvolvimento de um sistema.

A qualidade de um *software*, com efeito, está fortemente conectada com os defeitos que um sistema pode apresentar. Sem o teste, um sistema apresenta grandes chances de externar defeitos quando da execução de seus processos.

Assim, recomenda-se realizar o teste de *software* durante o desenvolvimento ou cada vez que se completa um ciclo do requisito do sistema, como a tela de cadastros. Para citar um exemplo; por meio dele, pode-se identificar uma forma de melhorar a qualidade do *software*, a confiabilidade, a usabilidade e a operacionalidade do sistema, além de evitar a continuação de *bugs*.

2.1 As Finalidades dos Testes de *Softwares*

Qual seria o objetivo precípuo em se submeter um determinado *software* a testes? Para Dias Neto (s.d, s.p.), "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". Caso não houvessem os testes, se um defeito fosse detectado no sistema, seria impossível saber qual a causa do mesmo. O aumento da qualidade do sistema, assim, é o que motiva a realização de testes.

2.2 O Funcionamento dos Testes de *Software*

O processo de teste se dá quando o desenvolvedor é informado por seu cliente que encontrou uma falha em seu *software*, ou quando determinada parte do *software* for finalizada, dando então, início a uma bateria de testes.

É importante estimar o tempo que um teste leva para ser executado, os recursos que serão utilizados, pessoas envolvidas, tecnologias necessárias, mas o

principal para o teste de software é que o analista responsável pelo teste tenha conhecimento do sistema, para que possa pensar em todas as formas possíveis de se executar determinado passo.

De acordo com Machado (s.d, s.p.), para se testar um sistema, leva-se em conta dois fatores, "probabilidade e risco".

Após a finalização dos testes, caso sejam encontrados erros, o analista deverá encaminhar para um desenvolvedor, para que este possa reparar a falha encontrada.

3 AS MODALIDADES DE TESTE

Todos os mecanismos desenvolvidos ao longo dos anos para o desenvolvimento de testes de *softwares* podem ser reunidos em quatro grupos: os testes de componentes, testes de integração, as caixas brancas e as caixas pretas. Tais mecanismos de avaliação dos sistemas merecem análise mais detida.

3.1 Testes de Componentes

Para se testar um *software* utilizando este tipo de teste, deve-se pensar no sistema em partes e testá-lo da mesma maneira. Nunes (s.d,s.p) diz: "testadores não devem testá-lo somente um sofware inteiro, mas, sim, visualizá-lo como uma parte do sistema e testá-lo separadamente" e depois afirma: "É necessário determinar as funcionalidades a serem testadas".

Para se testar determinada parte do sistema, antes deve-se garantir se é possível que aquele componente possa ser testado de maneira independente. É necessário ainda, ter a certeza de que somente as interações entre os componentes desejados irão ocorrer e que a substituição de um componente por outro não implicará no desejo de refazer testes em todo o sistema.

3.2 Testes de Integração

O teste de componentes se prova eficaz quando se diz respeito ao teste individual, porém Lourenço (s.d, s.p.) afirma: que este mesmo teste "não é capaz de garantir que as dependências funcionais entre componentes estejam perfeitamente implementadas." Surge então, a necessidade do teste de integração.

Podemos utilizar duas vertentes para aplicar o desenvolvimento de *software*, são eles: *Bottom-up* e *top-down*.

No primeiro, o teste inicia-se de baixo para cima, ou seja, com funções básicas até as de altas complexidades. Leciona Lourenço (s.d, s.p):

> Para realizar testes no desenvolvimento bottom-up, deve ser escrito código que invoque as rotinas de baixo nível, testando-as com diversas combinações de parâmetros. As rotinas escritas para tais testes são conhecidas como drivers, pois sua função é acionar o código que deve ser testado.

São criados códigos de programação para que possam executar diversos testes com parâmetros, ou seja, valores diferenciados para que ao final se consiga um *software* de qualidade.

Quando a abordagem de desenvolvimento utilizada é a *top-down* o teste é realizado de maneira inversa, de cima para baixo. Lourenço (s.d, s.p.) assevera ainda que "no teste up-down, a função inversa é fornecida pelos stubs que provêem dados para as rotinas de nível superior, permitindo que se realizem os testes".

Sendo assim, os testes são iniciados em funções de alto grau de complexidade até as funções.

Após testes e correções dos componentes de forma individual, inicia-se a construção de um novo código. No tocante à abordagem de desenvolvimento *bottom-up* os *drives* são substituídos por rotinas reais. Referente à abordagem top-down, os *stubs* continuam a fazer sua parte. As novas rotinas devem ser testadas novamente em conjunto e todo o processo se repetirá até que o software seja finalizado.

3.3 Caixa Preta

Outro teste útil para a assegurar a qualidade de software é o teste de caixa-preta. Nele, é examinado se a saída está de acordo com o resultado esperado, por exemplo, se uma rotina do sistema é responsável por fazer a soma entre dois números e o usuários insere o número um nos dois campos, espera que o resultado seja dois. No teste de caixa-preta, não há necessidade de conhecer o código por dentro do *software.* A respeito do teste de caixa-preta Lourenço (s.d,s.p.) leciona:

> O teste de caixa-preta é baseado nos requisitos funcionais do software. Como não há conhecimento sobre a operação interna do programa, o avaliador se concentra nas funções que o software deve desempenhar. A partir da especificação são determinadas as saídas esperadas para certos conjuntos de entrada de dados.

Este teste é útil, por exemplo, para casos de validações de campos, como máscaras de CPF, telefone e datas.

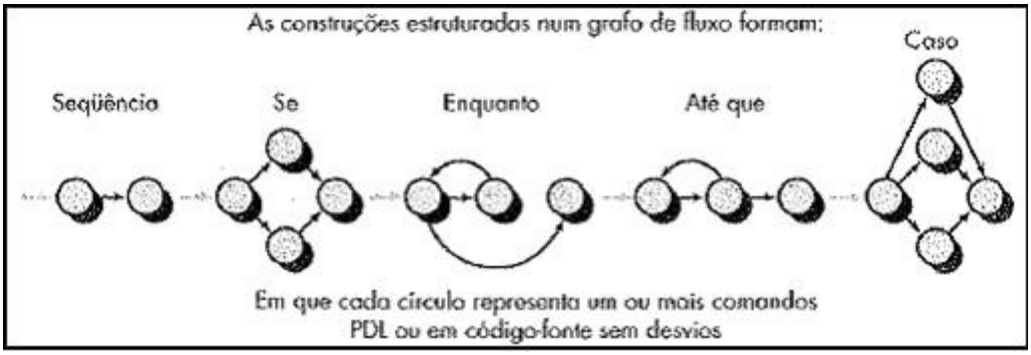
3.4 Caixa Branca

Diferente do teste de caixa-preta, neste, o responsável pela validação deve ter o conhecimento total do código em que realiza a avaliação, logo, para se realizar o teste de caixa-branca, o código fonte deve estar terminado.

Extrai-se do código fonte o grafo de fluxo de controle (GFC) para melhor interpretação da rotina do software. De acordo com Mccabe, citado por Cardilli (s.d, s.p) " O grafo de fluxo é um gráfico que demonstra a lógica do código fonte através de fios e ramos".

Os grafo de fluxo são construídos através de uma base principal de grafo e de acordo com Burnstein (203, p 709) *apud* Cardilli (s.d, s.p), pode ser constituído por "sequência, se, enquanto, até que e o caso".

Na imagem abaixo, pode-se ver um exemplo do grafo de fluxo utilizado para determinar caminhos na rotina de um *software*.



Representação base para o grafo de fluxo (Presmann, 2006).

Percebe-se então, que o teste de caixa-branca facilita ao desenvolvedor e ao analista de teste - em caso de erros - identificar qual a origem da possível falha que acometeu o *software*.

4 CONCLUSÃO

Diante do exposto alhures, conclui-se que para lançar um software não basta simplesmente desenvolvê-lo; existe a necessidade premente de testar o software utilizando-se de formas corretas de testes, sejam eles os testes de componentes, de integração, de caixa-preta e/ou caixa-branca.

Sem esses testes, o sistema possui altas chances de apresentar falhas nas execuções de suas rotinas, o que poderá comprometer a credibilidade do analista de sistemas e reduzir o valor do produto no já tão competitivo mercado da informática. Evitar tais ocorrências é justamente o que faz dos testes de *softwares* uma parte vital do desenvolvimento deles, consoante ficou latente ao longo do presente ensaio.

REFERÊNCIAS BIBLIOGRÁFICAS

CARDILLI, Danilo. **Uma visão da técnica de teste de caixa branca**. Disponível em: http://www.devmedia.com.br/uma-visao-da-tecnica-de-teste-de-caixa-branca/15610 acesso em 10 jun.2013.

LOURENÇO, Marcelo. **Teste de Integração**. Disponível em: http://qualidade-de-software.blogspot.com.br/2010/01/teste-de-integracao.html acesso em 10 jun. 2013.

MACHADO, Cristina Angela Filipak. **A importância dos testes no desenvolvimento de sistemas**. Disponível em: http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=883 acesso em: 09 jun. 2013.

MACORATTI, José Carlos. **Testes em Desenvolvimento de Software - Você Precisa Disto?** Disponível em: http://www.macoratti.net/tst_sw1.htm acesso em: 01 jun. 2013.

MERIAT, Vitor. **Testes e mais testes... O porquê dos testes de software**. Disponível em: http://vitormeriat.wordpress.com/2013/01/28/testes-e-mais-testes-o-porqu-dos-testes-de-software/ acesso em: 31 mai. 2013.

NETO, Arilo Claudio Dias. **Engenharia de Software - Introdução a Teste de Software**. Disponível em: http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035 acesso em 02 jun. 2013.

NUNES, Paulo Roberto de A. F. **Monografia:Teste de Componentes**. Disponível em: http://www.ime.usp.br/~prnunes/mac5766/monografia.pdf acesso em: 09 jun. 2013

PONTES, Melissa. **Verificação & Validação.** Disponível em: http://gotest.biz/wp-content/uploads/2009/06/introducao-a-testes-de-software_final.pdf acesso em: 01 jun. 2013.

PRESSMAN, R. S.**Engenharia de Software**. 6. ed. Rio de Janeiro: McGraw-Hill,2006, 720p.

REUTERS. **Testes de software movimentam US$ 16 bi por ano**. Disponível em: http://informatica.terra.com.br/interna/0,,OI501471-EI553,00.html acesso em: 01 jun. 2013.

SAMMY. **Estimando Esforço para Teste de Software**. Disponível em: http://www.testadores.com/index.php/more-about-joomla/432-estimando-esforco-para-teste-de-software acesso em: 31 mai. 2013.

SILVA, Fernando Rodrigues. **Testes de software - Entendendo Defeitos, Erros e Falhas**. Disponível em: http://www.devmedia.com.br/testes-de-software-entendendo-defeitos-erros-e-falhas/22280 acesso em: 29 mai. 2013.

SILVA FILHO, Antonio Mendes da. **Artigo Engenharia de Software 10 - Documento de Requisitos**. Disponível em: http://www.devmedia.com.br/artigo-engenharia-de-software-10-documento-de-requisitos/11909 acesso em: 31 mai. 2013

TOMAZ, Racquel. **5 Grandes mentiras sobre Teste de Software**. Disponível em: http://crowdtest.me/mentiras-teste-de-software/ acesso em 02 jun. 2013.