**Testes de Software – Testes Unitários**

**André Luiz da Silva¹**

¹Análise e Desenvolvimento de Sistemas – Faculdades Integradas Camões  
Alameda Dr. Muricy, 707 - 80020-040 - Curitiba – PR – Brasil

anluiz.designer@gmail.com

***Resumo:*** *Um documento que apresenta os principais processos de testes a ser realizado durante o desenvolvimento de um software, colocando em mais detalhes o Teste Unitário. Este processo é importante para controlar e validar áreas isoladas do código de um software, em outras palavras é a área mais baixa dos tipos de testes.*

# 1. Introdução

Os softwares no mundo atual já desempenha um importante papel para auxiliar nos negócios de empresas, e sua construção passa por diversas etapas, todas com sua devida prioridade e importância.

Sistemas atuais ainda devem conseguir ser integrados com outros sistemas já existentes, pois estão em constante evolução. Na crescente demanda nos dias atuais, boa parte dos negócios depende de software para manter o fluxo de trabalho, sendo assim a qualidade do sistema deve ser vital na sua execução. Um software de qualidade identifica quando recebe dados válidos e dados inválidos e age de maneira correta para cada situação.

Um software de qualidade deve passar por testes dentro do prazo e custos controlados e devendo ser compatível com o mercado, existem modelos que trazem ótimos resultados e são utilizados por grandes organizações, alguns destes modelos utilizados tem sido o ISO/IEC 12207, CMMI, SW-CMM. Testes de software é fundamental, porém é necessário ter habilidades, conhecimento e infraestrutura.

# 2. Histórico de Revisões

O presente documento reúne informações de diversas fontes, podendo ser melhorado ao longo do tempo.

|  |  |  |
| --- | --- | --- |
| **Versão** | **Data** | **Observação** |
| ANLZ V1.0 | 10-Agosto-2017 | Projeto iniciado com modelo de Artigo da SBC. |
| ANLZ V1.1 | 21-Agosto-2017 | Adição de conteúdo ao projeto. |
| ANLZ V1.2 | 30-Agosto-2017 | Adição de conteúdo de projeto. |

# 3. Visão Geral do Processo de Testes

Na década de 1970 o tempo dedicado na codificação e testes unitários eram maiores pela parte dos desenvolvedores. Isto era feito para garantir ao Usuário que o sistema estava funcionando de maneira correta, nessa época era considerado um mal necessário e não eram tratados como um processo formal alinhadas com o processo de desenvolvimento.

A partir da década de 1980, foi dada mais importância na análise de requisitos e com isso as atividades de testes ganharam mais importância e foi criado Metodologias de Testes que continuam evoluindo até os dias de hoje. Com a utilização da Internet para realização dos negócios, houve mudanças significativas nas aplicações, pois agora fatores tais como, segurança e desempenho passaram a ser relevantes tornando assim as atividades de testes mais especializadas.

# 3.1 O que é um Teste de Software?

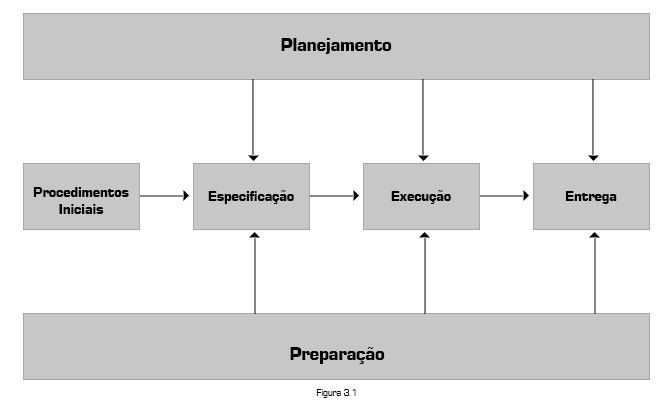
Existem diversas definições, mas na sua essência, teste de software é um processo feito de forma controlada para executar ações, com o objetivo de conferir o comportamento do que foi definido. Podemos dizer que é considerado como uma *validação*.

# 3.2 O processo de Testes

O processo de testes deve ser baseado em uma metodologia onde favorece o desenvolvimento, isto vai desde qualificação de pessoal, como também no ambiente e ferramentas adequadas.

A metodologia de testes deve ser um documento básico, da mesma forma que o desenvolvimento de um projeto é desejável haver uma metodologia adequada, o mesmo ocorre na área de testes.

A figura 3.1 apresenta o ciclo de fases de um processo de testes:



**Procedimentos Iniciais –** Elaboração do GOT (Guia Operacional de Testes), ou seja, um documento que contém um acordo entre as partes envolvidas (usuário, desenvolvimento, teste e produção), que define os assuntos e etapas do teste. Incluindo a responsabilidade de cada um, avaliação de riscos, níveis de esforço, entre outros.

**Planejamento –** Elaboração e revisão da estratégia e plano de teste.

**Preparação –** Preparação do ambiente de teste, incluindo equipamentos, pessoal, ferramentas e o software.

**Especificação –** Elaboração e revisão dos Casos de Teste.

**Execução –** Execução conforme elaborado o Caso de Teste.

**Entrega –** Conclusão do processo de teste, entregando o sistema para o ambiente de produção.

# 4. Tipos de testes

Adiante apresentamos a relação existente de testes, com suas respectivas definições e colocaremos neste documento com mais detalhes a respeito do Teste Unitário.

**Testes Caixa Preta (Black Box) –** Visa verificar as funcionalidades e comportamento do software visto pelo usuário ou alguém externo ao desenvolvimento, sem que eles tenham conhecimento do código ou da lógica do programa.

**Testes Caixa Branca (White Box) –** Avalia as lógica interna do componente, bem como as configurações e outros elementos técnicos.

**Testes Unitários –** Aplicado nos menores componentes dos códigos criados, visam garantir a funcionalidade e características conforme as especificações. Os testes unitários verificam uma parte isolada do sistema ou software afim de garantir sua funcionalidade. Este teste é na maioria das vezes realizado pelo próprio desenvolvedor.

**Testes de Integração –** Executados em uma combinação de componentes, que podem ser um código, módulos, aplicações distintas, etc. Este teste pode ser feito de forma incremental, onde cada módulo é adicionado conforme for sendo testado.

Este tipo de teste é exclusivo dos desenvolvedores ou equipe de testes.

**Testes de Sistema –** É realizado pela equipe de teste, buscando refletir o sistema no ambiente de produção mais próxima do real, utilizando cada função e módulos existentes. Aqui também é realizados os testes de cargas, performance, usabilidade, compatibilidade, segurança e recuperação.

**Testes de Aceitação –** Os testes finais onde é realizado pelos usuários, visando verificar se as soluções aplicadas atendem ao objeto de negócio e seus requisitos, mesmo assim a equipe de testes acompanham os usuários nesse período.

# 4.1 Outros tipos de testes

Além dos principais testes apresentado no tópico anterior, estes testes fazem parte dos processos, tais como *Testes de Regressão, Testes de Carga, Testes End-to-End, Testes de Configuração, Testes de Usabilidade, Testes de Instalação, Testes de Segurança, Testes de Recuperação, Testes de Compatibilidade, Testes de Desempenho, Testes Funcionais, Testes de Qualidade de Código, Testes de Alterações, Testes de Recuperação de Versões, Testes de Interoperabilidade, Testes de Sobrevivência, Testes Estáticos, Testes Embutidos*.

# 4.2 Testes Unitários

Testes unitários são realizados pelos desenvolvedores em si, por isso possivelmente eles mesmos definem os casos de testes. Como já dito, o teste unitário é feito para validar a funcionalidade e desempenho de um trecho do código, como por exemplo a execução de uma classe, função, método ou procedimento.

O Teste Unitário não é considerado uma fase do projeto, e sim algo realizado ao decorrer da evolução da construção software e seus componentes. Mesmo não sendo uma fase do projeto o Teste Unitário possui Requerimentos Unitários, dos quais são utilizados para encontrar possíveis problemas em determinada ação.

# 4.2.1 Recomendações nos Testes Unitários

Descreva os nomes dos testes com ‘o que’ e o ‘ porquê’ a partir da perspectiva do usuário. Se o desenvolvedor apenas de ler o nome do teste entenda o seu significado, então ele foi bem planejado. Veja o seguinte exemplo:

***<?php  
public class contaBancaria {***

***var $saldo;***

***function Conta($saldo){  
 $this->saldo = $saldo;  
 }***

***function temSaldo(){  
 if($saldo > 0){  
 return true;  
 }else{  
 return false;  
 }  
 }***

***}  
?>***

Como escrever um teste para esta classe?

***<?php  
class testeContaBancaria {***

***function testaContaComSaldo() {  
 $c = new contaBancaria(10.0);  
 if(c->temSaldo()){  
 echo “Ok”;  
 }else{  
 echo “Erro”;  
 }***

***}***

***$teste = new testeContaBancaria->testaContaComSaldo();***

***echo $teste;***

***?>***

O teste não finaliza por ai, pois e se alguém colocar um valor inicial 0 (zero) ou ‘NULL’? Então devemos ver todas as possibilidades de entrada e vai da imaginação do desenvolvedor planejar e executar elas. Veja como ficaria no final:

***<?php  
public class contaBancaria {***

***var $saldo;***

***function Conta($saldo){  
 $this->saldo = $saldo;  
 }***

***function temSaldo(){  
 if($saldo != NULL && $saldo > 0){  
 return true;  
 }else{  
 return false;  
 }  
 }***

***}***

***class testeContaBancaria {***

***function testaContaComSaldo() {  
 $c = new contaBancaria(10.0);  
 if(c->temSaldo()){  
 echo “Ok”;  
 }else{  
 echo “Erro”;  
 }***

***function testaContaSemSaldo() {  
 $c = new contaBancaria(0.0);  
 if(c->temSaldo()){  
 echo “Ok”;  
 }else{  
 echo “Erro”;  
 }***

***function testaContaNulo() {  
 $c = new contaBancaria(NULL);  
 if(c->temSaldo()){  
 echo “Ok”;  
 }else{  
 echo “Erro”;  
 }***

***}***

***$testeComSaldo = new testeContaBancaria->testaContaComSaldo();  
$testeSemSaldo = new testeContaBancaria->testaContaSemSaldo();  
$testeSaldoNulo = new testeContaBancaria->testaContaNulo();***

***echo $testeComSaldo . “<BR>”;  
echo $testeSemSaldo . “<BR>”;  
echo $testeSaldoNulo . “<BR>”;***

***?>***

Normalmente testes possuem mais linhas de códigos que a própria classe. Sempre busque fazer o teste no comportamento externo e não em sua estrutura e sempre utilize nomes que sejam de fácil entendimento conforme no exemplo acima.

# 4.2.2 Teste de Funcionalidade

Quando um teste unitário for bem realizado pode cobrir a necessidade de outros testes, como por exemplo o **Testes Caixa Preta (Black Box)** e o **Testes Caixa Branca (White Box)**, realizando os testes de valores nos inputs (passando parâmetros), retorno de valores corretos, performance do componente, entre outros. A seguir descreveremos testes unitários em Funções, Procedimentos, Classes e Métodos.

**Funções (Functions) –** Toda função retorna um valor. Por isso é a área mais fácil de testes unitários. Os casos de Testes de Funções derivam de uma combinação de parâmetros passados por todos os inputs. Todo o parâmetro passado pelo input, deve ser observado em sua saída (output), verificando detalhes técnicos como formato de datas, casas decimais, separadores, etc.

A boa prática para realização de testes unitários em functions é descrever como o valor será inserido, e qual a expectativa de retorno após a função ser acionada. Compare com o resultado apresentado e caso haja divergência com a expectativa, realize as alterações necessárias e repita o processo.

**Procedimentos (Procedures) –** Diferente de Funções, estes não retornam nenhum valor, porém ele faz alguma coisa. Eles podem enviar um sinal para ativar uma função em espera, podem enviar as informações recebidas para serem apresentadas em tela ou impressas ou alterar informações conforme a data do sistema, entre outros. Com isso percebe-se uma maior complexidade na execução do Teste de Caso.

A boa prática na realização de testes unitários em procedures é estar ciente de todo os efeitos que deveram ser acionados, e enviar as informações necessária para ativa-las. Fique de olho em cada atividade que a procedure realiza e desenvolver um glossário das atividades realizadas e quais irão realizar. Verificar dia a dia o glossário para poder acompanhar a procedure.

**Métodos (Methods) –** Métodos podem ser classificados como Função ou Procedimento, pois verificasse o resultado de acordo com os parâmetros de um input que geram uma atividade em espera. O que os diferencia é que um Método fornece um objeto quando o mesmo é requisitado pelo software.

A boa prática na realização de testes unitários em methods é verificar a visibilidade do mesmo para outros usuários, interno ou externo ao software. Caso o methods seja ativado sem a solicitação necessária, pode comprometer o desempenho e segurança do software.

**Classes (Class) –** Classes é um conjunto de métodos que interagem com a aplicação principal, o seu teste é feito como um todo, acionando todos os métodos existentes ou acionando apenas um método individualmente. O Caso de Teste de Classes é considerado uma forma de **Testes Caixa Preta (Black Box)**.

A boa pratica na realização de testes unitário em classes é verificar as propriedades inclusas na classe esse elas são acessíveis ou não via métodos SET e GET, dependendo da situação será gerado testes separados para validação. A aplicação principal do software deve ter comunicação com a classe, e os valores informados devem ativar o seu funcionamento, de acordo com as propriedades estabelecidas.

# 5. Conclusão

A aplicação de testes unitários na construção de um software pode trazer melhorias não somente aos clientes, bem como para os desenvolvedores, a qualidade do software e inclusive no relacionamento da empresa para com o cliente.

**Referencias**

[1] Young, C. R. Manfred; Galileo Computing Software Testing and Internationalization; Lemoine International, 2003.

[2] Moreira, T. R. Emerson; Teste de Software – 3ª Revisão revisada e ampliada; Alta Books Editora, Rio de Janeiro, 2013.

[3] Crespo, N. A., Silva, J. O., Borges, A. C., Salviano, F. C., Junior, A. T. M., Jino, M.; Uma Metodologia para Testes de Software no Contexto da Melhoria do Processo; Artigo Research Gate, 2015.

[4] Bogard, Jimmy; Getting values out of your unit tests; LostChies, 2008.

[5] Manual do PHP; <php.net/manual/pt\_BR>, Acessado em 12 de Setembro de 2017.