Melhores práticas para Testes de Software

Brenda Lee , Eduardo Aguiar

Camões - Faculdades Integradas

80020-040 - Curitiba - PR - Brasil

Disciplina Engenharia de Software

brenda.kikuta@gmail.com, eduardo.aguiar0@gmail.com

Abstract. *This document applies to best best practices for software testing, looking to the functionality, reduction of defects, search of the desired results, controlled execution of the software and evaluation of its behavior.*

Resumo. *Este documento aplica-se às melhores práticas para testes de softwares, visando sua funcionalidade, diminuição de defeitos, busca dos resultados desejados, execução de forma controlada do software e avaliação do seu comportamento.*

1. Fases das atividades de testes

1.1. Testes unitários

São os testes que abrangem as menores partes do programa, classes, métodos, funções, dentre outros.

1.2. Automação de teste unitário

É o uso de uma ferramenta para gerenciar o teste do software, geralmente é programado manualmente para executar a automação sempre que implementado ou alterado algum trecho do código.

1.3. Teste de integração

Feito após o teste unitário, o teste de integração é responsável pela construção da estrutura do sistema.

1.4. Teste de sistema

Verifica se as funcionalidades especificadas nos documentos de requisitos estão corretamente implementadas.

1.5 Teste de regressão

Este teste é feito somente após uma manutenção do sistema, ou seja, só será feito caso seja necessário. Para que não corra o risco de lançar a nova atualização com erro, é feito o teste de regressão, onde é voltados os passos e testados novamente, garantindo assim, a maior eficiência do software.

2. Técnicas das utilizadas nas Melhores práticas

2.1. Caixa branca

Identifica os erros de programação, estruturas, lógica e laços. A sua função é testar unidades do software e seu processo realizado durante todo o desenvolvimento do mesmo.

2.2. Caixa preta

Neste método, não interessa os códigos que estão no desenvolvimento do software, e sim dados de entrada e saída.

Possibilita identificar erros de interface, acesso aos dados de um banco, desempenho e velocidade da aplicação.

É realizado no fim do desenvolvimento do software, mas também pode ser utilizado durante o processo.

2.3. Caso de testes

O que interessa é o que foi obtido no final, assim é gerado um documento com a especificação das entradas e os resultados que são esperados das Entradas x Saídas de trechos do código.

3. Utilizando UML como ferramenta para casos de testes

A UML (Unified Modeling Language) é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. O objetivo do uso da UML para testes de software, junto com o padrão IEEE829, provê a validação dos requisitos implementados. O desenvolvimento dos casos de uso propostos seguem os seguintes padrões:

- ➔ Ler a descrição, o fluxo principal e alternativos do caso de uso;
- ➔ Identificação das pré-condições e pós-condições;
- ➔ Desenvolvimento das especificações de teste de cada um dos cenários;
- ➔ Identificação das variáveis operacionais para o cenário principal;
- ➔ E por último, desenvolver os procedimentos para execução dos testes.

4. Entendendo as práticas para testar seu software

Para a efetivação das melhores práticas deve-se cumprir todos os passos desde o momento da construção do software.

Neste modelo os testes são feitos intercalados com o desenvolvimento, dividido em duas partes: programação e teste. Será utilizado o modelo de Verificação e Validação (modelo em V).

Ao final de cada etapa é iniciada a bateria de testes, gerando o relatório de erros e considerações pertinentes para que se inicie o processo de correção do software.

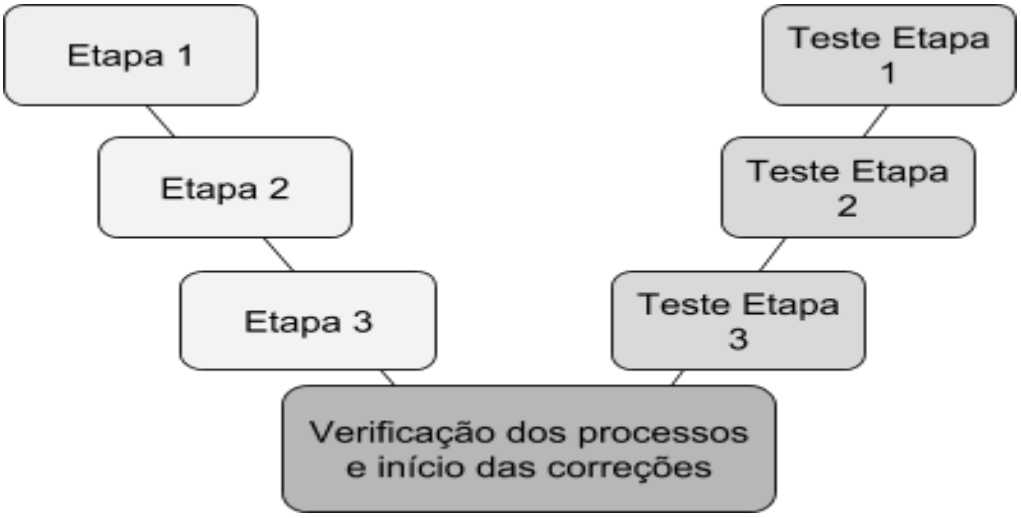

Figura 1. Modelo de Verificação e Validação

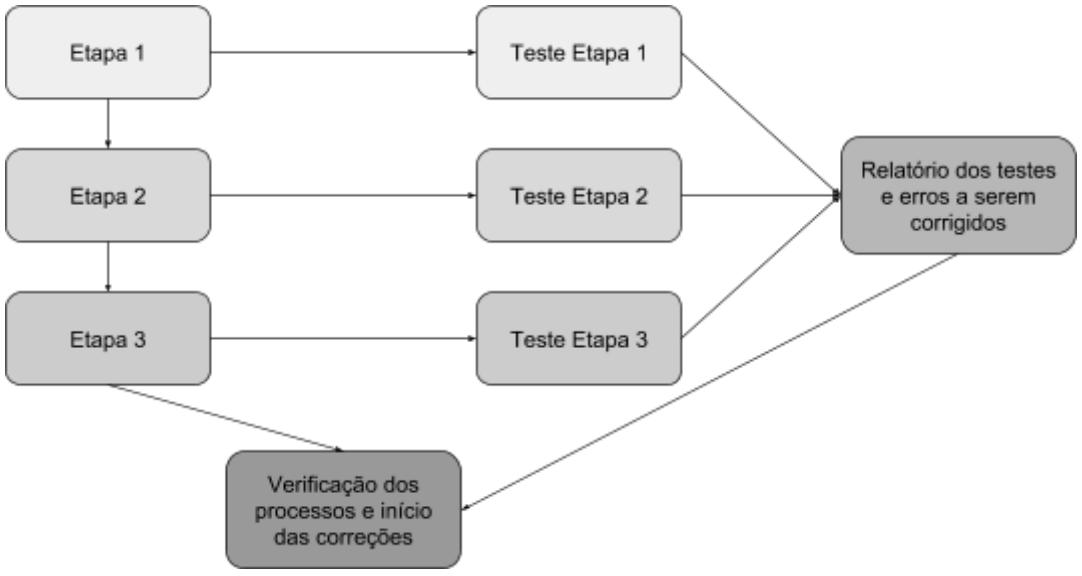

Figura 2 - Diagrama de processo

5. Colocando em prática

Para cada teste é feito um checklist dos passos que devem ser executados para chegar ao êxito do software, cada teste tem sua própria característica (citadas no começo do documento) e passos a seguir.

Teste unitário	❑ Identificar o método, classes, objetos ou partes do código a ser testado ❑ Executar uma determinada função do código ❑ Se for um método de cadastro efetuar um teste de inserção ❑ Identificar se o valor de saída é o esperado
Automação de teste unitário	❑ Identificar qual ferramenta será utilizada para o teste ❑ Teste de caso: ❖ Teste de design ❖ Preparação para teste da data base ❑ Teste da data base: ❖ Executar código com o teste de data base ❑ Testar resultados obtidos ❖ Comparar resultados com teste de caso ❑ Gerar resultados finais
Teste de integração	❑ Identificar os módulos que serão combinados no teste ❑ Testar individualmente os módulos ❑ Testar agrupando os componentes ❑ Testar a integração dos componentes ❑ Identificar erros de lógica ou fluxo de dados
Teste de sistema	❑ Identificar se o sistema está no escopo do teste caixa preta ❑ Testar expectativa do cliente ❑ Identificar aceitações positivas ou negativas
Teste de regressão (apenas se for necessário)	❑ Efetuar a atualização do sistema ❑ Utilizar teste de automação ❑ Identificar novos defeitos ❖ Regredir a versão do sistema ou manter versão atualizada

6. Relatório de correções

Ao final das etapas em que o software está completo e todos os testes foram efetuado, é gerado um relatório de erros a serem corrigidos, inicia-se então o processo de correção e todo a prática de Teste de Software é refeita a cada passo modificado.

Lembrando que isso não é um Teste de regressão, seria um "Teste de validação e correção", pois só será necessário o Teste de regressão caso haja algum imprevisto em que o projeto deve voltar em um ponto por motivos maiores.

Assim que os novos testes retornarem sucesso em todos os requisitos significa que as Melhores práticas para Testes de Software obtiveram êxito no seu processo e pode ser liberado para as novas etapas do projeto.

7. Considerações

O teste de software é uma prática extremamente promissora, gera informação a fim de corrigir e/ou até prevenir erros, quer seja para a boa organização de uma empresa, elaboração e melhoria de processos, ou até, senão mais importante, para tomada de decisões.

Referências

[Emerson Rios], [Trayahú Moreira] "Teste de Software - 3° Edição revisada e ampliada",

https://books.google.com.br/books?hl=pt-BR&lr=&id=I2a2BAAAQBAJ&oi=fnd&pg=PA125&dq=o+que+s%C3%A3o+teste+unitarios+de+software&ots=uzlZHq8vMr&sig=GxbV87N8JfvX5lUz_CDj70cePvs#v=onepage&q=o%20que%20s%C3%A3o%20teste%20unitarios%20de%20software&f=false, Agosto

[Márcio Eduardo Delmaro], [José Carlos Maldonado], [Mario Jino] Introdução ao Teste de Software - 2° Edição,
https://books.google.com.br/books?hl=pt-BR&lr=&id=1rA4DwAAQBAJ&oi=fnd&pg=PR1&dq=testes+de+integra%C3%A7%C3%A3o+de+software&ots=qEYx5_RU6a&sig=JqP1cphhCJIwfnfUzwUxxccQo_I#v=onepage&q=testes%20de%20integra%C3%A7%C3%A3o%20de%20software&f=false, Agosto

[Davi do Amaral], [Renato Gomes], [Rodrigo Passos de Jesus], [Tiago Marques de Araujo], [Elias E. Goulart] Metodologias de Teste de Software,
http://www.ria.net.br/index.php/ria/article/view/38/38, Agosto