A Importância do Processo de Teste de Software em TI

Mateus Bruno Teixeira Lopes [1], Allan Guerreiro Carneiro[2].

[1]Faculdade de Ciências Biológicas e da Saúde - Tecnólogo em Redes de Computadores
Av. Maria de Paula Santana, nº 3815 Bairro Silvestre - Viçosa, MG - 36570-000

[2]Universidade Federal de Viçosa, Viçosa, MG – 36570-000

mateus_redes@hotmail.com, allanguerreiro@gmail.com

Abstract. *This article describes the importance of the process of software test for companies of technology of the information, with objective of showing the techniques, types, phases, tools, models and norms that involve the documentation and execution of the software tests obtaining like this an increase in the quality of the produced software.*

Resumo. *Este artigo descreve a importância do processo de teste de software para empresas de tecnologia da informação, com objetivo de mostrar as técnicas, tipos, fases, ferramentas, modelos e normas que envolvem a documentação e execução dos testes de software obtendo assim um aumento na qualidade do software produzido.*

1. Introdução

A atividade de teste de software é uma área da TI que vêm crescendo ao longo dos anos e que está diretamente relacionada à necessidade de se produzir produtos de qualidade que atendam a exigências cada vez maiores. Segundo Myers “Testar um programa é analisar um programa com a intenção de descobrir erros e defeitos”, podemos citar também, a definição dada por Dijkstra “O teste de programas pode ser usado para mostrar a presença de defeitos, mas nunca para mostrar a sua ausência”.

Percebemos que as empresas que produzem software vêm sofrendo cada dia mais pressões de mercado/clientes/usuário que sempre exigem produtos de alta qualidade, são pouco tolerantes com atrasos nas entregas, fazendo com que as empresas possuam precisem de softwares e equipes mais complexas.

Tendo em vista atender estas necessidades, existem muitos sistemas em que uma falha ocorrida pode causar prejuízos diretos ou mensuráveis como, por exemplo, a empresa Symantec proprietária do software antivírus Norton que em junho de 2007 teve de compensar 50 mil vítimas de uma atualização que retirava arquivos de sistema de uso, dando a elas uma extensão de 12 meses da licença do Norton e uma cópia da ferramenta Norton Save & Restore 2.0. Já, em outros sistemas, as falhas podem causar prejuízos imensuráveis tal como o de, 1988, quando o navio US Vicennes derrubou um Airbus 320 causando a morte de 290 pessoas, a falha foi atribuída ao software de reconhecimento do navio que confundiu o avião com um F-14.

Mesmo precisando atender a tantas exigências e correndo tanto risco, por que as empresas não testam o software que produz?

- Muitas vezes, o software é muito complexo e está sempre sendo modificado;
- A maioria dos projetos e softwares são desenvolvidos sob crescente pressão para entrega em prazos rigorosos sem ter tempo de checar as atividades realizadas;
- Há um desconhecimento sobre o custo x benefício dos testes;
- Há falta de profissionais especializados ou qualificados para apresentar as técnicas e implantar um processo de teste adequado à organização;
- O teste lida com pessoas e muitos têm a ideia de que os testadores são inimigos dos desenvolvedores porém isto é um mito, pois o trabalho em equipe e a interação dos testadores e desenvolvedores serão muito úteis para um aumento da qualidade do software produzido;
- Não sabem o percentual e o alto custo do re-trabalho, alem de não perceberem que é muito mais viável realizar testes do que ter de corrigir uma falha ocorrida no software em fase de produção;

- Não sabem quais são as áreas de produção que geram mais re-trabalho e com isso não conseguem tomar iniciativa para corrigir as áreas mais ineficientes;

- Às vezes se preocupam em testar apenas no final do projeto, ocasionando a necessidade de alteração de todo o produto, ou parte do mesmo, gerando assim um alto custo de re-trabalho.

2. Qualidade de Software

Todo software visa atender a uma demanda, seja ela questão de qualidade, economia, confiabilidade, demanda, negócio e outras, desse modo, para garantirmos a eficiência do programa é que devemos testar o software, pois é provável que existam defeitos, através dele também podemos reduzir custos e descobrir algumas características e qualidades existentes num software tais como: confiabilidade, qualidade, desempenho, usabilidade, portabilidade, segurança, recuperabilidade e outras.

Quando se propõe um processo de teste de software a uma organização é importante considerar que cada uma tem o seu processo de desenvolvimento de software já planejado e criado, assim para a implantação desse novo processo devemos respeitá-lo e fazer com que o novo seja o mais integrado possível ao existente no entanto, que sejam independentes, podendo ser iniciados e realizados em paralelo por equipes qualificadas e especializadas que farão verificações para identificar se o sistema foi construído corretamente alem de, validações respondendo se o sistema foi construído corretamente ou não.

Uma das formas de aproximar o processo de teste de software com o processo de desenvolvimento é o modelo “V” que focaliza no teste durante todo o ciclo de desenvolvimento do software. A Figura 1 exemplifica o modelo.

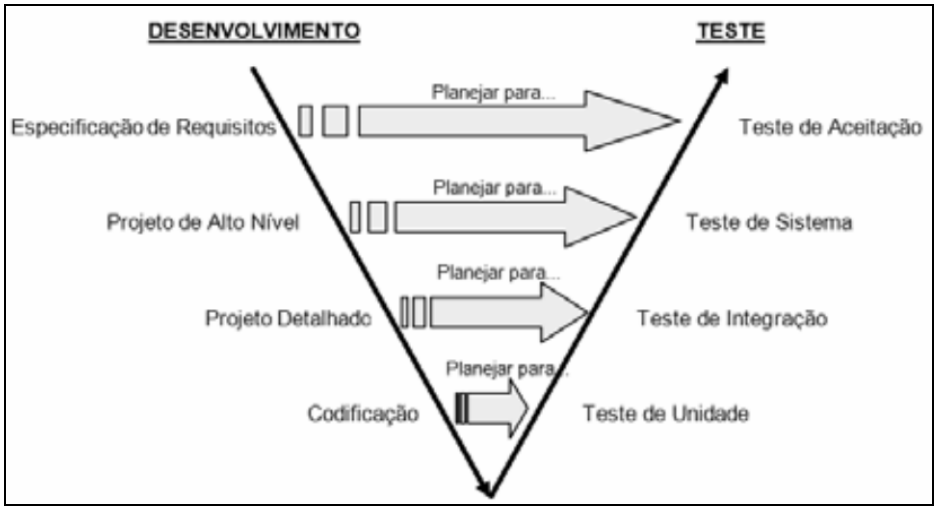

Figura 1: Representação do modelo "V" como aproximação do processo de teste de software com o processo de desenvolvimento do software (GRAIC e JASKIEl, 2002).

Fonte: Revista Engenharia de Software Magazine, 2007.

3. Tipos de Testes

Os tipos e técnicas de teste podem ser classificados em:

- **Caixa branca**: esta é uma técnica de teste geralmente realizada pelos desenvolvedores que trabalha diretamente com o código-fonte do sistema.
- **Caixa preta**: esta é uma técnica de teste geralmente realizada pelos analistas de teste, testadores e usuários, que trabalha com a interface do sistema, sem se preocupar com o código-fonte.

Como representado na figura 1, o modelo "V" vem para integrar os processos de desenvolvimento e teste do software desta forma, podemos citar os quatro níveis de teste que definem em que momento deve ser testado o software.

- Testes Unitários: Testam se cada funcionalidade especificada na codificação e desenho do software foi implementada

corretamente, são geralmente realizados por desenvolvedores, utilizando a técnica de Caixa branca, como ferramenta para automação destes testes podemos citar o JUnit que auxilia muito em teste de sistemas Java.

- Testes de iteração ou integração: Tipo de teste realizado após o teste unitário, quando se tem um software que faz a interação com outro(s) produto(s) seja ela da mesma desenvolvedora ou de terceiros faz-se estes testes para garantir que a comunicação entre estes dois produtos seja implementada como especificado na codificação, desenho e arquitetura do sistema. Este nível de teste pode ser realizado tanto por desenvolvedores como por testadores e é seguido por dois métodos de interação:
    - Abordagem Top-Down: Abordagem que funciona bem com desenvolvimento estruturado que auxilia mais cedo na identificação de problemas no desenho da arquitetura mais cedo.
    - Abordagem Bottom-Up: Abordagem que funciona bem com desenvolvimento orientado a objetos que só conseguem identificar os maiores problemas de arquitetura tardiamente.
- **Testes de Sistema:** Tipo de teste realizado por testadores após todos os testes de integração, pois valida o sistema como um todo podendo analisar os requisitos funcionais e não funcionais, como desempenho, volume, documentação, robustez. Para automatização dos testes de sistemas existem ferramentas como a TestComplet e o JUnitPerf.
- **Testes de Aceitação:** Teste realizado por testadores que possuem responsabilidade também dos clientes, sendo possível automatizar estes testes com as ferramentas JMeter e JUnitPerf, pois, auxiliam na verificação do software para que o mesmo entre ou não em produção.

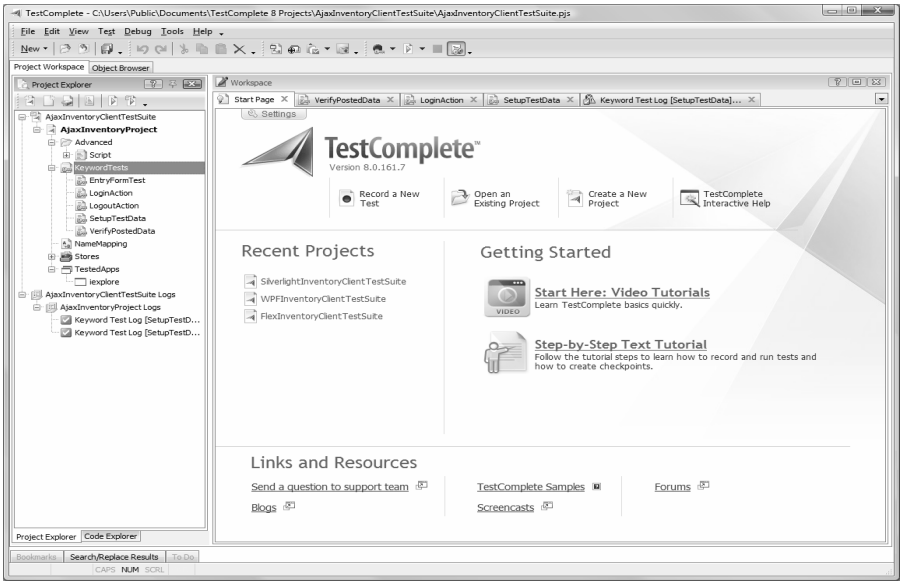

Figura 2: Utilização da ferramenta TestComplete para automação dos teste de software. Fonte:. http://www.automatedqa.com, 2010.

4. Características de Qualidade

A Norma ISO 9126-1 define seis características de qualidade que o software deve atender, sendo elas:

- Funcionalidade: verifica a capacidade do sistema em prover funcionalidades definidas que atendam as necessidades do usuário, quando usado sob determinadas condições pré-estabelecidas.

- Confiabilidade: o produto de software é capaz de manter seu nível de desempenho, ao longo do tempo, nas condições estabelecidas de utilização.

- Usabilidade: capacidade do software em ser entendido, aprendido e utilizado sob condições estabelecidas de utilização.

- Eficiência: os recursos e os tempos envolvidos são compatíveis com o nível de desempenho requerido pelo software.

- Manutenibilidade: refere-se ao esforço necessário para a realização de

alterações específicas no produto de software.

- Portabilidade: facilidade de o software poder ser transferido de um ambiente para outro.

5. Técnica de Teste Estruturado

Estas características estão ligadas ao tipo de técnica de teste utilizada no processo, sendo dividida em duas como a técnica de teste estrutural que tem o objetivo de verificar se o software está estruturado e funciona corretamente, e a técnica de teste funcional que tem o objetivo de verificar se os requisitos e as especificações foram atendidos assim, essas duas técnicas possuem tipos de testes específicos para utilização das mesmas.

1. Técnica de Teste Estruturado
    - Teste de estresse: Realizado para verificar o comportamento do sistema em condições-limite como condições de hardware precárias ou grande volume de dados.
    - Teste de contingência: Realizado para verificar se o sistema consegue se recuperar de uma falha ou, caso ocorra, seja possível retomar o serviço.
    - Teste de segurança: Realizado para garantir que apenas pessoas autorizadas possam acessar dados/informações/interfaces a elas destinadas, analisando vulnerabilidades e ameaças ao sistema tanto físicas como lógicas.
    - Teste de performance/desempenho: Realizado para medir tempos de respostas, volume de dados suportados, números de transações e outros dados que interferem no desempenho do sistema.
    - Teste de conformidade: Realizado para garantir que o software esta de acordo com o projeto, processos e suas especificações.
2. Técnica de Testes Funcionais
    - Teste de funcionalidade: Realizado para garantir que o software estaja de acordo com as funções implementadas e atenda aos requisitos definidos.

- Teste de regressão: Realizado para garantir que as novas funcionalidades não alterem o funcionamento das outras partes do sistema que não sofreram alteração.
- Teste de interconexão: Realizado quando se tem a integração com sistemas externos ou de terceiros.
- Teste de usabilidade: Realizado para verificar se o software está de fácil entendimento e manuseio pelos usuários teste muito importante, pois muitos softwares não são projetados para usuários com necessidades especiais e isso faz com que estas pessoas não consigam utilizar o seu produto, o que poderia ser um diferencial.

Esses tipos, técnicas e fases de testes existentes definem qual estratégia utilizaremos em nosso processo de teste nossa estratégia deve conter também quando, o que e como vamos testar determinado software a fim de estabelecer quais partes serão testadas ou inspecionadas bem como suas prioridades de testes, por meio da análise de riscos do negócio que deve ser realizada em consideração ao nível de impacto ou classificação do prejuízo que o risco pode causar ao projeto, e a probabilidade ou classificação da chance de um risco se concretizar.

A definição da estratégia de teste de software é muito importante para o processo de teste, pois, ela vai influenciar na montagem do plano de testes, na qualidade do software e nos testes que serão realizados, afinal a definição da qualidade de um software pode variar de acordo com a quantidade de defeitos encontrados e a qualidade dos testes feitos.

6. Plano de Testes

Elaborar um plano de teste é uma tarefa que exige envolvimentos dos usuários, desenvolvedores, testadores, apoio da gerência, treinamento/capacitação dos envolvidos, caso seja necessário, pois, por ser uma documentação considerada grande, ela precisa ser feita com calma analisando todas as etapas de sua elaboração, podendo seguir um dos três principais e mais utilizados padrões existentes como o padrão PMI,

padrão QAI e o padrão IEEE da Norma 829 ambos são padrões a serem seguidos, porém, cada organização pode fazer adaptações ou criar o seu próprio modelo para utilização.

O padrão IEEE 829 apresenta a seguinte estrutura para o plano de testes:

- Identificação do Plano de Testes
- Referências
- Introdução
- Funcionalidades a serem testadas
- Riscos do processo de teste
- Funções a serem testadas da perspectiva do usuário
- Funções que não serão testadas da perspectiva do usuário
- Abordagem (estratégia de testes)
- Critérios de conclusão dos testes
- Critérios para interrupção e retomada de testes
- Entregáveis
- Ambiente de Testes
- Pessoal e Responsabilidades
- Cronograma
- Plano de riscos
- Aprovação do teste

Ao término da elaboração de seu plano de testes, a Norma IEEE 829 descreve três documentos que fazem parte também da fase de documentação da preparação dos testes contendo os seguintes documentos:

- Especificação do Projeto de Teste;
- Especificação de Casos de Teste;
- Especificação de Projeto de Teste.

Ao término da elaboração de toda esta documentação os próximos passos definidos pela norma é a execução dos testes seguindo as documentações anteriores durante a execução, com certeza serão encontrados defeitos que devem ser

documentados ou registrados através dos seguintes documentos:

- Diário de Teste/ Log de Teste;
- Relatório de Incidentes de Teste;
- Relatório Resumo de Teste;

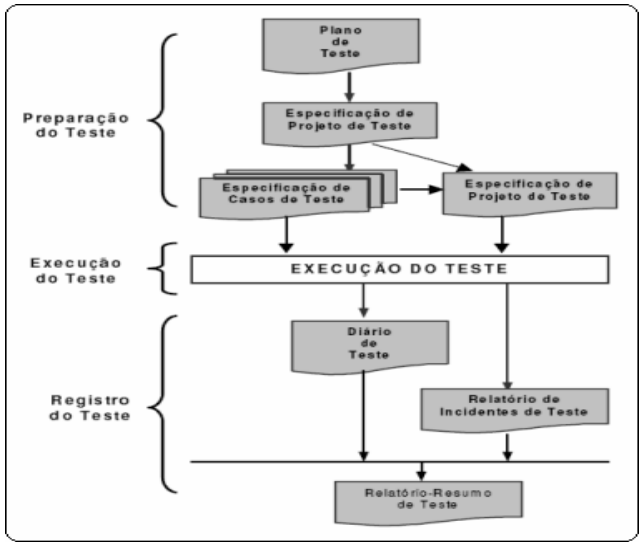

Figura 3: Processo de Teste de Software. Fonte: Norma IEEE 829.

Podemos também utilizar ferramentas como Bugzilla, Mantis e Jira para armazenar e gerenciar tais defeitos encontrados, sempre lembrando que a documentação e reportagem de um defeito encontrado devem ser feitas de forma clara e detalhadas para que o analista ou desenvolvedor consiga reproduzir o mesmo, tornando assim mais fácil e rápida sua correção.

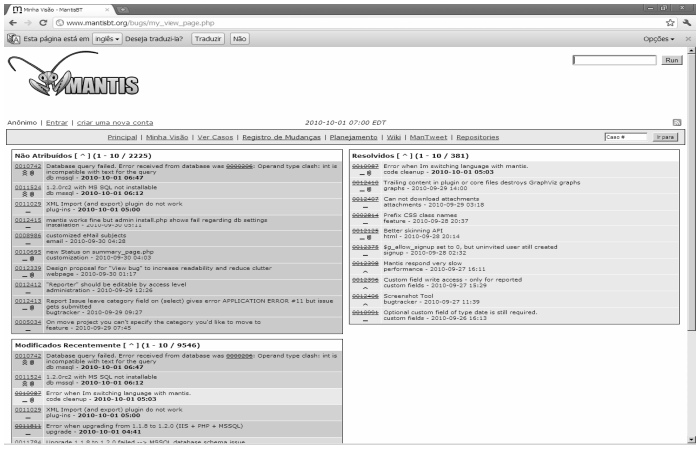

Figura 4. Ferramenta Mantis para gerenciamento dos defeitos encontrados. Fonte: http://www.mantisbt.org, 2010.

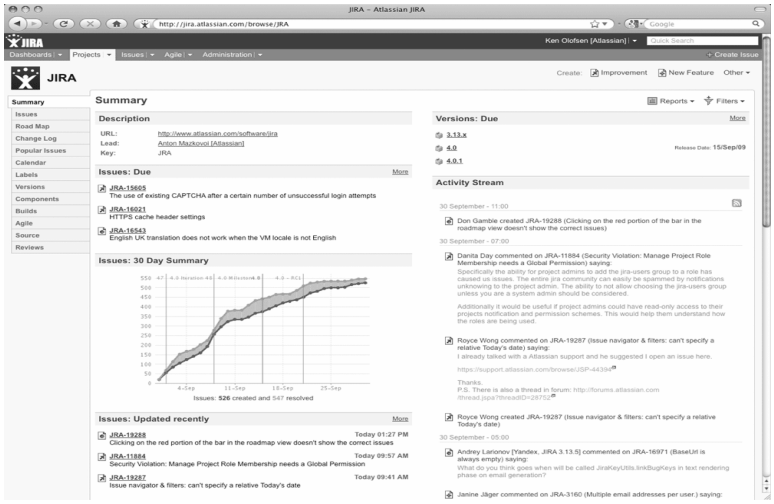

Figura 5. Ferramenta Jira para gerenciamento dos defeitos encontrados.
Fonte: http://www.atlassian.com, 2010.

7. Considerações finais.

A utilização do processo de teste de software não deve ser feita apenas por que tem de ser feita, mas sim porque sabemos que testar faz com que a qualidade, a credibilidade, a confiança e a competitividade do software cresçam.

Em muitos casos os programas são testados isoladamente à medida que os módulos vão sendo concluídos, a fim de confirmar que o módulo foi codificado corretamente. Depois, grupos de programas são testados num "teste de sistema" onde é feito um teste de integração para testar as interfaces e assegurar que os módulos estão se comunicando da maneira esperada. Em seguida, o software é explorado como forma de detectar suas limitações e medir suas potencialidades. Em um terceiro nível, sistemas completos são, por fim, submetidos a um "teste de aceitação" para verificar a possibilidade de implantação e uso, geralmente feita pelo cliente ou usuário final.

É fundamental em todos os ramos da engenharia de software garantir a produção de software de alta qualidade a fim de proporcionar aos usuários uma maior confiança e segurança na utilização do mesmo.

8. Referências.

[Revista, 2007] "Revista Engenharia de Software Magazine", Ano 1, 1º Edição 2007.

[TIEXAMES, 2010] "Curso de Fundamentos em Teste de Software", concluído em 27 de setembro de 2010.
www.tiexames.com.br

[JUnit, 2010] "JUnit". Acessado em 30 de setembro de 2010.
http://www.junit.org

[JMeter, 2010] "Apache JMeter". Acessado em 30 setembro de 2010.
http://jakarta.apache.org/jmeter

[TestComplete, 2010] "AutomatedQA TestComplete". Acessado em 1 de outubro de 2010.
http://www.automatedqa.com

[Jira, 2010] "Jira", Acessado em 30 de setembro de 2010.
http://www.atlassian.com

[JUnitPerf, 2010] JUnitPerf. Acessado em 30 de setembro de 2010.
http://www.clarkware.com/software/JUnitPerf.html

[Mantis, 2010] "Mantis Bug Traking System". Acessado em 1 de outubro de 2010.
http://www.mantisbt.org

[TestExpert] "TestExpert – A sua comunidade de teste e qualidade de software". Acessado em 1 de outubro de 2010.
http://www.testexpert.com.br