Faculdade de Tecnologia de Americana
Curso de Processamento de dados

A IMPORTÂNCIA DO TESTE NO PROCESSO DE PRODUÇÃO DO SOFTWARE

THIAGO SECOMANDI GALDINO

Americana, SP

2010

Faculdade de Tecnologia de Americana
Curso de Processamento de dados

A IMPORTÂNCIA DO TESTE NO PROCESSO DE PRODUÇÃO DO SOFTWARE

THIAGO SECOMANDI GALDINO

furbyy.thiago@gmail.com

Trabalho de graduação apresentado à Faculdade de Tecnologia de Americana, como parte dos requisitos para obtenção do título de Tecnólogo em Processamento de Dados.

Área: Teste de Software

Americana, SP

2010

BANCA EXAMINADORA

Prof. Alberto Martins Jr (Orientador)

Prof. José Renato de Siqueira Lopes

Prof. Antonio Alfredo Lacerda

Dedicatória

A meus pais, amigos e mestres que me apoiaram e motivaram a subir mais um importante degrau na minha formação pessoal e profissional.

Agradecimentos

A Deus, pela vida.

Aos meus amigos por todo o apoio e incentivo, principalmente nos momentos mais difíceis.

Ao Professor Alberto Martins Júnior, pela orientação, conselhos e atenção na realização deste trabalho.

À IBM e aos meus companheiros, que sempre me deram oportunidades de agregar novos conhecimentos e aplicá-los em meu crescimento profissional.

Aos amigos que conquistei na FATEC, agradeço pelos momentos compartilhados e espero que nossos caminhos não se distanciem a ponto de tornar-nos apenas lembranças um para o outro.

“Não viva para que sua presença seja notada,
mas para que sua falta seja sentida”

Robert Nesta ‘Bob’ Marley

Resumo

Cada vez mais a tecnologia tem tomado espaço em nossas vidas, seja nas mais corriqueiras situações ou em nossos ambientes de trabalho, podemos vê-la em todo lugar. Hoje em dia se fala muito de segurança e usabilidade de programas, e para alcançar níveis aceitáveis de qualidade, todo produto deve passar por um controle, que segue padrões e metodologias pré-definidas.

Este trabalho visa mostrar que o Teste de Software está diretamente ligado à satisfação do cliente e / ou usuário através da garantia de que todas as funcionalidades da aplicação estão executando suas ações de forma a seguir os requisitos antes do lançamento do produto.

PALAVRAS CHAVE: Testes; automação; ferramentas de teste; aplicação.

Abstract

Increasingly, technology has taken place in our lives, either in the most mundane situations or in our workplaces, we can see it everywhere. Nowadays, everybody talks about security and usability of these applications, and in order to achieve acceptable levels of quality, every product must go through a control process, which follows pre-defined standards and methodologies.

This article meant to present the Software Testing is directly linked to customer and / or user satisfaction by ensuring before the product launch that all the features in the application are performing their actions following the requirements.

KEYWORDS: Testing; automation; test tools; application

Lista de Figuras

Lista de Abreviaturas e Siglas

IEEE - Institute of Electrical and Electronics Engineers

ISTQB – International Software Testing Qualifications Board

NBR - Denominação de norma da Associação Brasileira de Normas Técnicas (ABNT)

SQA - *Software* Quality Assurance, em português Garantia da Qualidade de *Software*

URL - Uniform Resource Locator, em português Localizador de Recursos Universal.

Web - World Wide Web, sistema de documentos em hipermídia que são interligados e executados na Internet

Sumário

Introdução

O IEEE (Institute of Electrical and Electronics Engineers) oferece uma definição de teste de software no Glossário de Terminologias de Engenharia de Software (IEEE Standard 610.12-1990):

"O processo de operar um sistema ou um componente sob condições especificadas, observando e registrando os resultados, e fazendo uma avaliação de alguns aspectos do sistema ou componente."

(Craig e Jaskiel 2002) propõem uma definição expandida e elaborada: "Teste é um processo concomitante do ciclo de vida da engenharia, usando e mantendo os artefatos de teste a fim de medir e melhorar a qualidade do software sendo testado."

Em outras palavras, teste significa comparar "o que é" com "o que deveria ser" (Copeland, 2004), ou seja, avaliar a capacidade de um programa de gerar os resultados requeridos, com o objetivo principal de identificar os defeitos presentes nos sistemas de software. Por isso o teste de software é normalmente considerado uma atividade destrutiva.

Objetivos

O objetivo deste trabalho é apresentar:

- A importância dos testes e algumas metodologias utilizadas para garantir que o software atenda às necessidades do usuário final;
- Uma introdução ao processo de automação de testes e como este pode ser aplicado com mais eficiência para auxiliar nas atividades tidas como repetitivas;
- Apresentar algumas ferramentas gratuitas utilizadas para gerenciamento e execução de testes manuais e automatizados;

Justificativa

Testes são desde os primórdios o principal médoto de verificação de qualidade e segurança de aplicações desenvolvidas para suprir as necessidades de um usuário final, também chamado de ‘cliente’.

Este estudo propõe uma pesquisa onde são apresentados modelos de trabalho para tornar a atividade de testes mais eficaz e, conseqüentemente, fazer com que os softwares cada vez mais possuam um valor elevado de confiabilidade e usabilidade.

Organização

Além deste capítulo introdutório, a organização do presente trabalho encontra-se da seguinte forma:

No capítulo 2, “Teste de Software”, são apresentados conceitos gerais da atividade de testes de software, mostrando os tipos de testes, suas fases, e uma visão geral do que são ferramentas de teste .

O capítulo 3, “Relevância da Atividade de Teste”, é dedicado a apresentar

efetivamente a relação direta entre a boa qualidade dos testes e a boa qualidade do software a ser testado, mostrando metodologias que podem ser utilizadas nesse processo.

No capítulo 4, “Automação de Testes”, são apresentados os conceitos básicos de automatizar um teste, quando isso é ou não efetivo e uma visão geral das melhorias que esta prática traz ao projeto.

No capítulo 5, Conclusão, são apresentadas as conclusões deste trabalho e sua contribuição para o conhecimento de testes em aplicações Web.

No apêndice A é apresentada uma relação de ferramentas *open source* usadas no gerenciamento e execução de testes manuais.

No apêndice B é apresentada uma lista de ferramentas *open source* utilizadas para execução de testes automatizados e de performance.

2. Teste de *Software*

Teste é uma forma de verificação que faz uso de técnicas e estratégias com o objetivo de executar a aplicação simulando o máximo de situações possíveis e determinar se ela se comporta de acordo com o especificado.

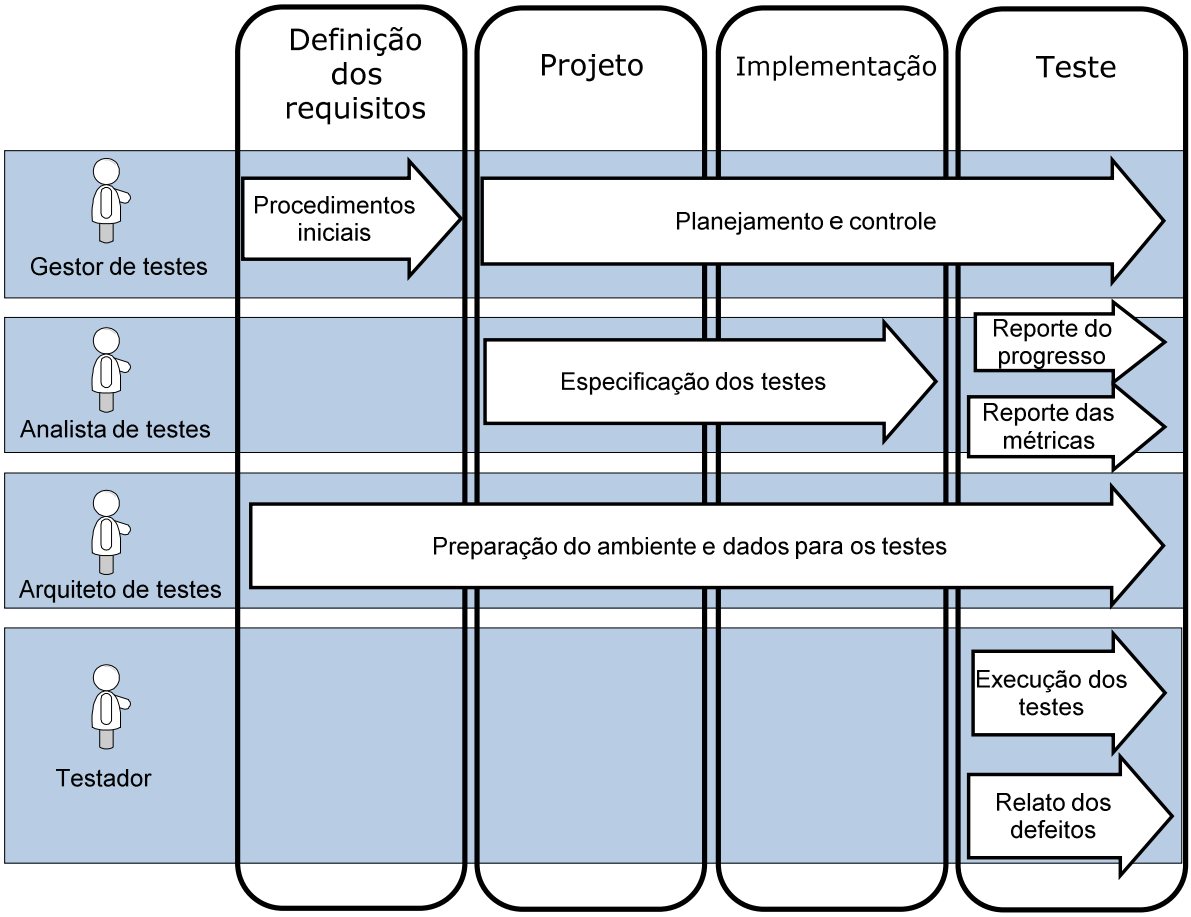
(MYERS, 1979) enumera alguns itens que podem servir como objetivos do teste:

- Teste é um processo de execução de uma aplicação com a finalidade de encontrar um erro.
- Um bom caso de testes é aquele que tem alta probabilidade de encontrar um erro ainda não descoberto.
- Um teste bem-sucedido é aquele que descobre um erro ainda não descoberto.

A atividade de teste engloba ainda dois processos que são a Verificação e a Validação. Verificação é o conjunto de atividades que garante que a aplicação executa corretamente uma função específica. Validação se refere ao conjunto de atividades que garante que o *software* construído corresponde aos requisitos do cliente.

Em outras palavras, a Validação responde as seguintes questões: ‘desenvolvi uma aplicação que é a que esperam?’, ‘Entendi perfeitamente o que o usuário espera desta aplicação?’, ‘O usuário sabe qual será a aplicação que será entregue?’. Ou seja, ‘A aplicação atende aos requisitos do usuário/cliente?’. Na Verificação é determinado se o *software* foi construído corretamente, se foram utilizadas as técnicas de desenvolvimento de *software* como modelagem, especificação e programação.

Figura 1 – Processo de Gestão de Testes e seus Personagens



Fonte: (CAETANO, 2007)

2.1. Testes Caixa-Preta

Também conhecidos como funcionais, são aqueles baseados nos requisitos, não interessando qual a estrutura interna da aplicação. Os testes são projetados visando encontrar discrepâncias entre o que a aplicação faz contra as suas especificações, (PRESSMAN, 2002-2006) e (MYERS, 1979).

O nome caixa preta advém do fato que não se considera como o *software* está construído, não se avalia o que ele tem por dentro. Para a especificação das entradas e saídas esperadas dos testes é levada em consideração a especificação. Deve-se preocupar com as funcionalidades e características do *software* e não com os detalhes de sua implementação.

- Testes de Performance

Teste que verifica certos parâmetros de performance da aplicação como: tempo de resposta e disponibilidade do serviço solicitado. Geralmente simula centenas de usuários a acessar a aplicação num determinado intervalo de tempo. Essas informações são analizadas, estimando assim os níveis de carga, para os quais existe um esgotamento dos recurso. As falhas apontadas através do teste são devidas ao ambiente de execução como: recursos escassos, mal distribuídos ou ainda oscilações na rede.

- Teste de Carga

Diferente dos testes de performance na medida uma vez que sua medição é a partir de um nível de carga pré-definido. Mede o tempo necessário para executar diversas ações sobre uma configuração minima e uma configuração máxima de atividade da aplicação. São simulados diversos navegadores a executar a aplicação, essas informações são gravadas e sempre que há uma falha, são gerados relatórios com as condições em que esse erro ocorreu.

- Teste de Stress

Avalia o comportamento do sistema num estado limite (ou superior) segundo a sua especificação. Usado para avaliar a resposta do sistema em picos de atividade que excedem os limites, verificando assim se o sistema *"*quebra*"* ou se é capaz de se recuperar de tais condições.

- Teste de Compatibilidade

Determina se a aplicação é executada como esperado num ambiente que contém diferentes condições de *hardware* e *software*. Devido à enorme diversidade de ambientes e componentes existentes na Web é impossível que todas elas sejam testadas. Usualmente, apenas as combinações mais comuns são verificadas, tendo

como consequência, que apenas um subconjunto de possiveis falhas seja descoberto.

- Teste de Usabilidade

Os testes de usabilidade, são de extrema importância, pois é a melhor forma de verificar se um sistema corresponde aquilo que é pretendido. Apesar de existirem ferramentas que verificam as interfaces segundo um conjunto de normas de usabilidade, estas não são 100% confiáveis.

Podem ser realizados dois tipos principais de testes de usabilidade:

As realizadas pelo próprio programador, verificando se uma série de requisitos de usabilidade foram implementadas de forma eficaz.

E por um grupo de usuários (seguindo o perfil definido inicalmente para o usuário tipico das aplicações), que segue um guia de ações pré-definidas, sendo os seus erros, tempos de resposta, expressões e comentários escritos e muitas vezes gravados.

- Teste de Acessibilidade

Verificar se uma aplicação Web segue um conjunto de normas definidas como as melhores práticas de acessibilidade para um grupo de usuários (por exemplo, pessoas com deficiências auditivas, visuais, motoras, cognitivas…). Normalmente são feitos manualmente, seguindo um *check-point list,* podendo-se verificar a implementação destas boas práticas. Existem ferramentas no mercado que podem automatizar este tipo de testes, porém não é dispensado o teste manual.

- Teste de Segurança

Avalia a eficácia de resposta da aplicação contra usuários não autorizados ou acessos indevidos e a capacidade da aplicação de permitir aos seus usuários o acesso aos serviços e recursos diponibilizados pela mesma. Também podemos enquadrar neste grupo os testes de penetração

- Sql Injections;
- Key Loggers;
- ...

2.2. Testes Caixa-Branca

Os testes caixa branca são também conhecidos como testes estruturais ou ainda baseados na implementação, focam as estruturas internas do *software*. Eles são projetados no código do programa e têm como objetivo exercitar seus caminhos, condições, ciclos e estruturas de dados. Os testes caixa branca são mais utilizados nas fases iniciais do teste principalmente nos testes de unidades e em menor escala nos testes de integração, (PRESSMAN, 2002-2006).

- Teste Unitário

Teste onde uma pequena porção da aplicação é testada de cada vez. Necessitam de ser executados no lado do servidor como no lado do cliente (browser):

No componente cliente é aquele que contém a interface gráfica (links, texto, imagens, navegação). Por vezes também inclui validação de input e funções simples;

No componente servidor é aquele que implementa a lógica de negócio da aplicação, coordenando a execução dos diferentes pedidos e controlando

o armazenamento e selecção de informação para uma central de arquivos de informações (BD, XML,TXT...).

- Teste de Integração

Testa um conjunto de páginas da aplicação, verificando como elas funcionam em conjunto.

A documentação, mostrando as relações entre as páginas, pode ser usada para definir grupos de páginas que podem ser testadas em conjunto. Pode-se dar como exemplo considerar páginas ligadas entre elas, como uma unidade a ser testada. Deve ser considerada tanto o comportamento como a estrutura da aplicação.

- Teste de Sistema

Tenta encontrar defeitos relacionados com toda a aplicação, que devem incluir:

Cobertura de funções e casos de uso do usuário;

Cobertura de páginas da aplicação (tanto cliente (browser) como servidor web);

Cobertura dos links

2.3. Teste em Aplicações Web

Nas aplicações Web, Pressman apresenta uma abordagem que adota os princípios básicos para o teste de todo *software* e aplica estratégias e táticas que são recomendadas para sistemas desenvolvidos para Web:

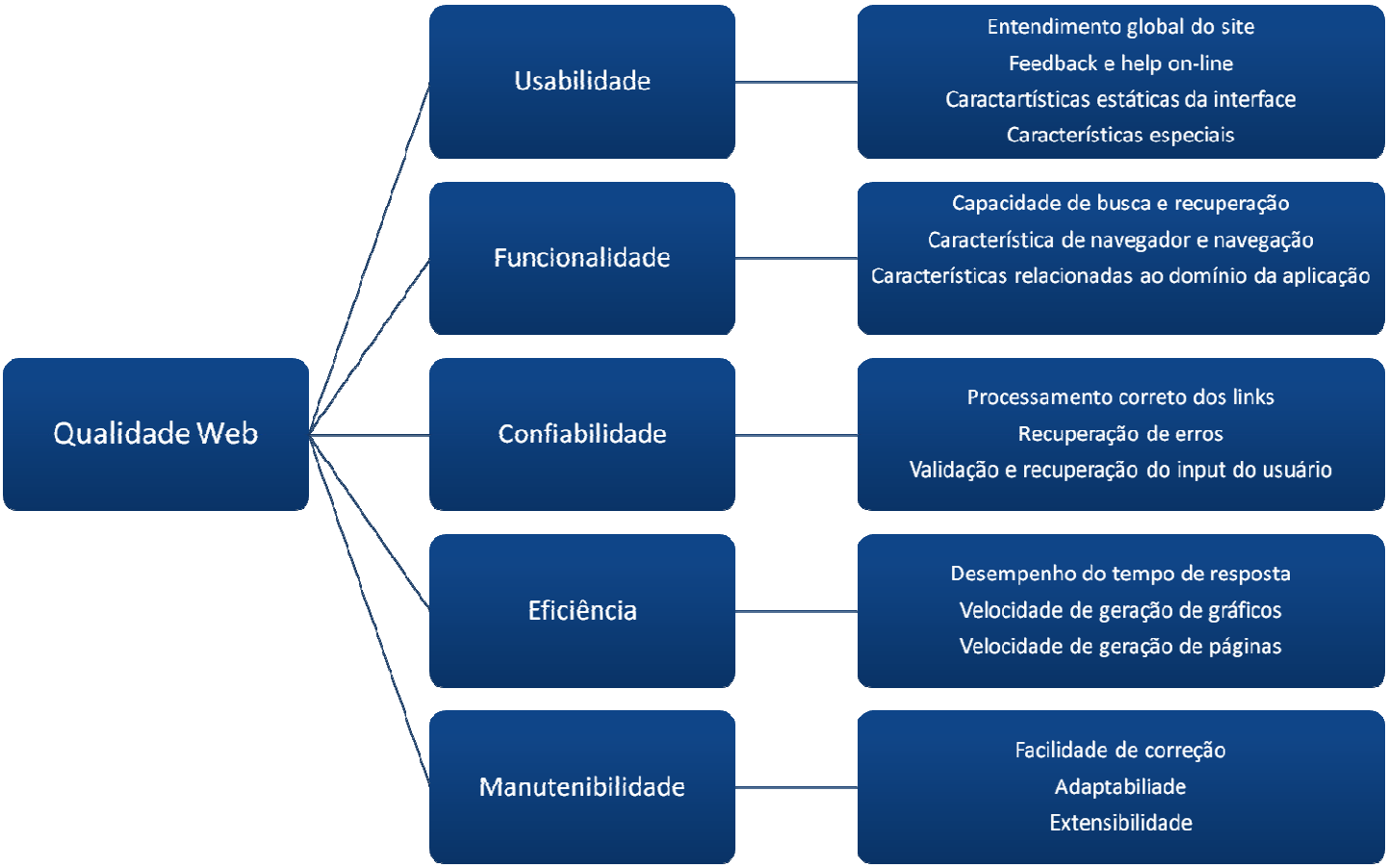
- Revisar o conteúdo a fim de descobrir erros de grafia, gramática, consistência do conteúdo, representações gráficas, dentre outros;
- Conferir o projeto de navegação para descobrir erros nos *links* e/ou na permissão de acesso de usuário;
- Testar cada página focando seu processamento;

- Confirmar a funcionalidade geral e conteúdo fornecido;
- Testar a compatibilidade da aplicação em diferentes configurações (diferentes *browsers*, sistemas operacionais, plataformas de hardware e protocolos de comunicação).

Ainda Pressman afirma que um “teste completo não é possível”. A quantidade de diferentes caminhos possíveis dentro de uma aplicação assim como a variação de entradas é muito grande, o que torna praticamente impossível cobrir todas as combinações de uma aplicação ou todas as suas entradas.

Existem basicamente duas maneiras de se construírem testes baseados na execução: método de caixa branca e método de caixa preta.

Figura 2 - Árvore de Qualidade de Aplicações Web



Fonte: (PRESSMAN, 2002-2006)

2.4. Ferramentas de Testes

As ferramentas de planejamento, controle e desenvolvimento de software são essenciais para um projeto, e as de teste também são. (Loveland, 2005) faz um paralelo entre as ferramentas de teste e as ferramentas de um carpinteiro para exemplificar a necessidade e a influência das ferramentas em um projeto: da mesma forma que um carpinteiro não consegue executar seu trabalho sem as suas ferramentas, um testador também não consegue executar propriamente seus testes. Por outro lado, não importa quão boas são as ferramentas do carpinteiro, o que irá decidir a qualidade do projeto é sua habilidade e experiência na execução do trabalho, além de seu conhecimento e prática com as ferramentas. O mesmo ocorre com as ferramentas de teste de software.

É importante ressaltar que a adoção de uma ferramenta não assegurará a qualidade do produto ou do processo. Uma só ferramenta não é suficiente para executar todas as tarefas do projeto, e sim uma coletânea delas. Isto é conhecido como “mito da ferramenta mágica”. Se as ferramentas forem adequadas e de qualidade, o trabalho do testador será facilitado, agilizado e organizado.

No apêndice “A” deste documento, foi listado algumas ferramentas de gerenciamento e execução de testes classificadas como *open source.*

3. Relevância da Atividade de Testes

As fábricas de software e prestadoras de serviço, principalmente as grandes empresas que trabalham com clientes internacionais e governamentais, vêm adotando cada vez mais os processos de testes. Em parte pela conscientização dos profissionais, mas principalmente pela crescente exigência de certificados de qualidade por parte dos clientes, que estão se conscientizado da necessidade de um ciclo eficaz de testes. Mas antes de entender o teste de software é necessário compreender por que ele é necessário.

Os sistemas de informação não são mais apenas ferramentas para melhorar a produtividade e eficiência de uma organização. O uso da tecnologia da informação chegou a um ponto no qual pequenas empresas, grandes corporações e até mesmo o governo simplesmente não podem mais sobreviver sem o uso de computadores e sistemas de informação (Loveland, 2005).

Como todo o ciclo de vida de desenvolvimento de software depende diretamente de seres humanos, e os seres humano são passíveis de cometer erros, defeitos podem ser inseridos durante a análise, codificação e implantação de sistemas computacionais. Esses defeitos podem gerar falhas, e essas falhas podem causar muitos problemas como perda de dinheiro e tempo, além de denegrir a reputação de uma empresa, e chegando, até mesmo, a causar danos físicos e mortes. Por outro lado, falhas também podem ser geradas por mudanças no comportamento do hardware, causadas por condições climatológicas externas ao escopo do desenvolvimento do sistema (ISTQB Syllabus, 2005).

Testes rigorosos do sistema e da documentação podem ajudar a reduzir significativamente o risco decorrente da ocorrência de falhas em ambientes operacionais, auxiliando no processo de correção de falhas, e contribuindo, assim, para a qualidade do sistema de software.

A atividade de teste tem ganhado relevância no processo de software. Alguns clientes exigem que um processo de testes seja utilizada no desenvolvimento de seu

produto de software. Com interesse de satisfazer o cliente, algumas empresas estão buscando a criação de Centros de Especialidade em teste e a certificação de seus profissionais.

O ISTQB (International Software Testing Qualifications Board) é uma organização internacional sem fins lucrativos cujo objetivo é fomentar o reconhecimento das disciplinas de teste e qualidade de software como parte da especialização profissional nesta área (ISQTB, 2006). O ISTQB oferece exames de certificação, auxiliando a definir um padrão para o desenvolvimento de uma carreira em testes de software, o que motiva profissionais da área e demonstra o crescimento, importância e reconhecimento das atividade de teste de software.

3.1. Metodologias

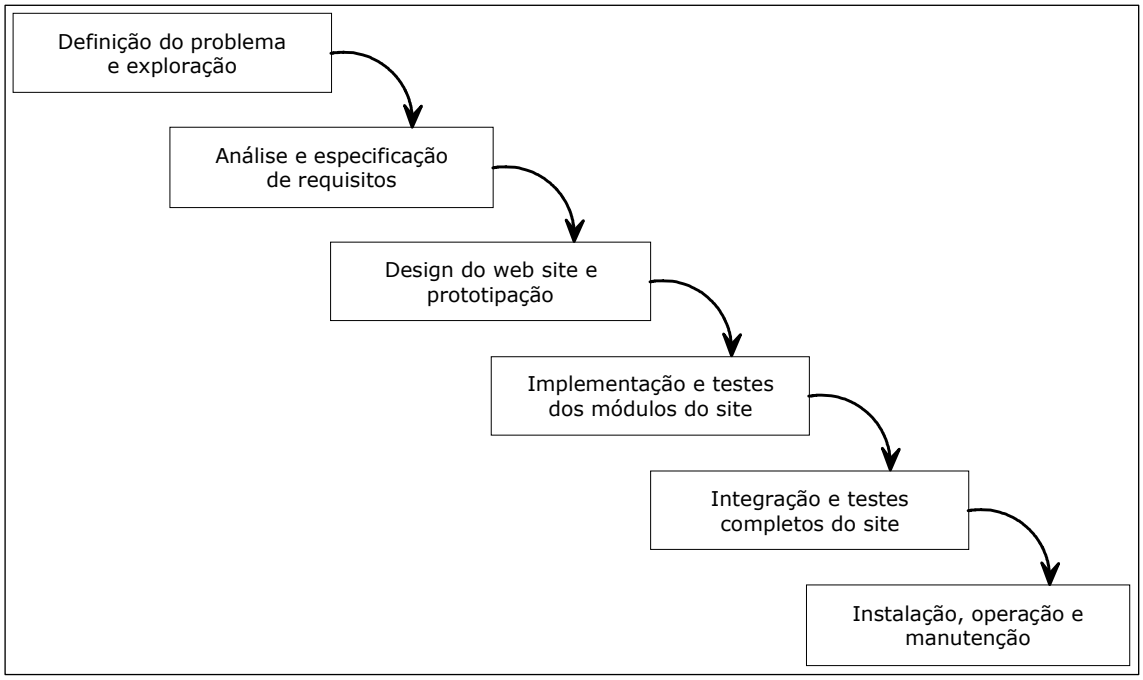
Há diversos processos diferentes que podem ser adotadas no processo de produção de um *software*. Dependendo do tipo da aplicação, tempo de duração do projeto e recursos disponíveis para todo o período de desenvolvimento e testes, varia também a melhor metodologia a ser utilizada.

Hoje em dia as metodologias mais comuns são a de cascata e *Agile*, que serão descritas mais detalhadamente abaixo.

3.1.1. Metodologia Cascata

É a mais difundida e utilizada para o processo de criação de aplicações Web e também para aplicações batch por possuir uma ordem cronológica que obedece a uma lógica simples, consistindo em testar a aplicação como um todo ao final do processo de desenvolvimento e modificar somente as funcionalidades onde forem encontrados defeitos.

Figura 3 – Modelo Cascata



Fonte: (LEITE, 2002)

Nesta forma de trabalho, os testes são desenvolvidos e desde a definição dos requerimentos e construídos durante o desenvolvimento da aplicação, sendo que qualquer mudança nos requisitos deve ser reportada, documentada e causará impacto nos testes já construídos, sendo necessária a revisão e correção dos mesmos.

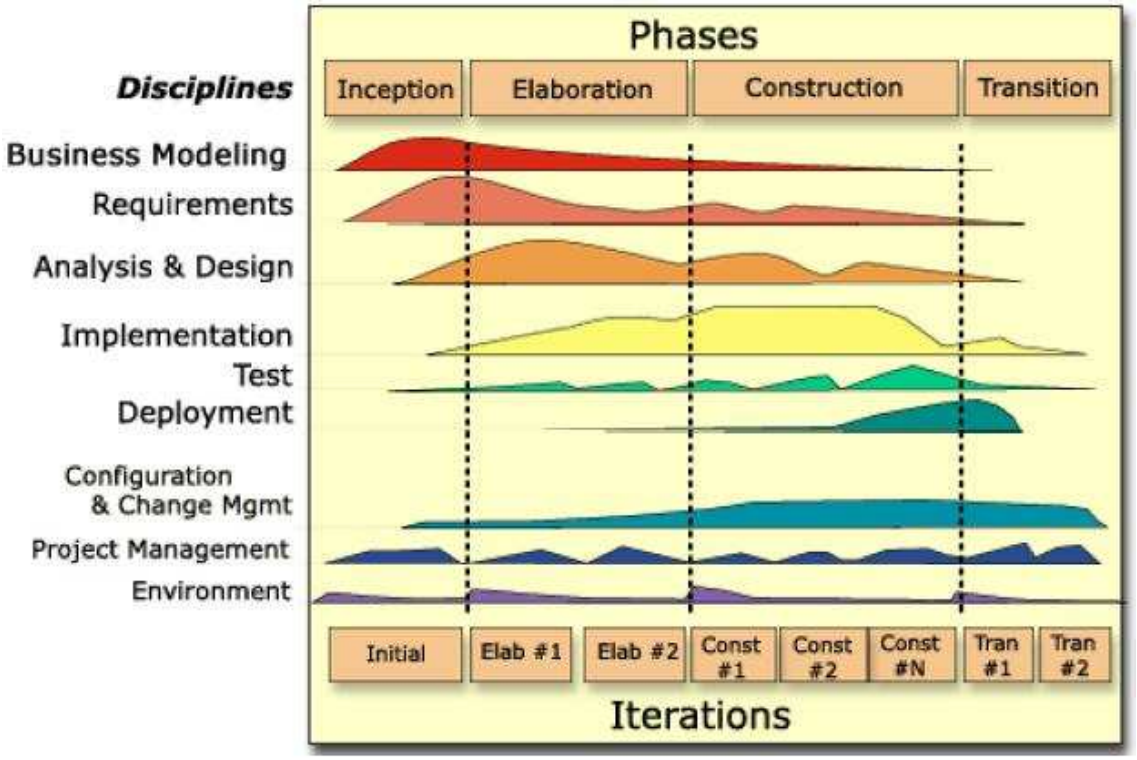
3.1.2. Metodologia *Agile*

Ultimamente essa metodologia vem tomando espaço no mercado, tomando uma grande fatia de empresas que querem tornar seus projetos mais ‘ágeis’ como o nome da metodologia sugere, mas percebem que se não for seguida a risca e com grande afinco, essa metodologia não traz os resultados esperados.

Esta prática consiste em ter grupos de trabalho de 2 pessoas, para que uma só funcionalidade não sobrecarregue um único recurso, e reunir todas as informações sobre o andamento do projeto em uma reunião diária chamada de *Scrum Meeting*, que consiste em 15 minutos de conversa onde cada um dos funcionários abora os seguintes tópicos:

- O que foi feito ontem?
- O que será feito hoje?
- Há algo que impeça que suas tarefas de hoje sejam completadas?

Figura 4 – Diagrama de atuação de recursos na metodologia *Agile*



Fonte: *(PACHECO, 2008)*

4. Automação de Testes

Nos últimos tempos, a automação de testes tem se tornado uma atividade vital em projetos de teste. Esta melhoria consiste em repassar para o computador casos de teste que seriam realizadas manualmente, trabalho maçante e bastante susceptível a erros de quem os realiza. Imagine alocar 1000 pessoas (cada uma em uma máquina) para realizar um teste de carga numa aplicação?

A automação de teste é uma das melhores formas de reduzir o tempo de teste no ciclo de vida de uma aplicação, diminuindo o custo e aumentando a produtividade no desenvolvimento como um todo e, conseqüentemente, aumentar a qualidade do produto final. Estes resultados podem ser obtidos principalmente na execução do teste de regressão, que se caracteriza pelo teste de aplicações já estáveis que passam por uma correção de erros, ou de aplicações já existentes que ao serem acrescidas para uma nova versão e suas funcionalidades são alteradas.

Os testes mais indicados para serem automatizados são aqueles que necessitam de um processo repetitivo (por exemplo, testar o login na aplicação com vários usuários) e aqueles que são executados com mais freqüência durante o processo de criação da aplicação.

Há testes que não possuem a condição de serem automatizados, como aqueles que envolvam algum tipo de intervenção física, ou que seja um teste complexo e com possibilidade de que um erro de menor importância ocorra durante sua execução, mas sem afetar o resultado final do teste.

No apêndice “B” deste documento, foi listado algumas ferramentas de automação de testes e testes de performance classificadas como *open source.*

Conclusão

Os testes têm sido cada vez mais importantes no processo de criação dos softwares, pois garantem a qualidade, confiabilidade e a segurança do mesmo. Através de boas práticas de teste, garantimos a qualidade de um software, qualquer seja seu objetivo.

O objetivo deste estudo foi apresentar a importância da atividade de teste, sejam eles automatizados ou não, as metodologias que podem ser aplicadas e algumas ferramentas para gerenciar testes, bem como executá-los manual ou automaticamente.

Entretanto, por ser uma atividade pouco valorizada, muitas vezes o processo de testes é ‘sobreposto’ de alguma forma e os prazos que atrasam durante o desenvolvimento da aplicação geralmente recaem sobre os testadores, encurtando o tempo hábil para a verificação e correção de defeitos, tornando o processo de testes uma prática muitas vezes deficiente em qualidade, o que pode trazer grandes prejuízos futuramente, nas fases de implementação e manutenção da aplicação.

Bibliografia

CAETANO, C. **Automação e Gerenciamento de Testes - Aumentando a Produtividade com as Principais Soluções Open Source e Gratuitas**. e-Book, 2007.

COPELAND, L. **A Practitioner's Guide to Software Test Design**. Ed. Artech House, 2004.

CRAIG, R. D., JASKIEL S. P. **Systematic Software Testing**. Ed. Artech House, 2002.

HUNG, N. Q. **Testing Applications on the Web**. Wiley Computer Publishing; 2001.

LEITE, J. C. **Desenvolvimento e design de sistemas Web**. Natal, RN. 2002. <http://www.dimap.ufrn.br/~jair/>. Acesso em: 20 de maio de 2010.

LOVELAND, Scott. **Software Testing Techniques: Finding the Defects that Matter.** Ed. Charles River Media, 2005.

MOLINARI, L. **Testes De *Software* - Produzindo Sistemas Melhores e Mais Confiaveis**. Érica. 2003.

MYERS, G. **The Art of *Software* Testing**. John Wiley & Sons, 1979.

PETERS, J. F.; PEDRYCZ, W. **Engenharia de *Software*: Teoria e Prática**. Rio de Janeiro, RJ. Campus. 2001.

PRESSMAN, S. R. **Engenharia de *Software***. Rio de Janeiro, RJ: Makron Books, 2002-2006.

RICCA, F.; TONELLA, P. **Analysis and Testing of Web Applications**; Proceedings of the 23rd International Conference on *Software* Engineering (ICSE'01). 2001

Apêndice A

Ferramentas de Gerenciamento e Execução de Testes Open Source

Ferramenta: QATraq **Descrição:** QATraq (2006) é uma ferramenta web para gerenciamento de testes. Suporta gerenciamento de requisitos e de procedimentos de teste, organização de planos de testes e execução manual. Possui diversos tipos de relatórios, além de rastreamento de requisitos e progresso de execução dos testes. O controle de defeitos é simples, mas pode ser integrado a uma ferramenta externa mais completa, como o Bugzilla. Realiza controle de versão de maneira transparente.
Ferramenta: TestLink **Descrição:** TestLink (2006) é uma ferramenta web para gerenciamento de testes. Auxilia no processo de criação de requisitos e procedimentos de teste, na organização dos testes em planos de testes, e na execução manual destes planos. Os resultados podem ser acompanhados pela interface e através da geração de relatórios que facilitam a visualização do progresso, dos defeitos e requisitos relacionados ao testes. Pode ser integrado a uma ferramenta externa para rastreamento de defeitos. Mantido por uma comunidade aberta de testadores.

Ferramenta: Salomé-TMF

Descrição: Salomé Test Management Framework (2006) é uma ferramenta bastante semelhante à ferramenta Test Director (2006). Possibilita a criação e gerenciamento de procedimentos, planos e dados de teste. Sua arquitetura é orientada a plug-ins, o que possibilita a adição de funcionalidades de maneira facilitada. Por exemplo, através de plug-ins, permite integração com ferramentas de teste automatizado, rastreamento de defeitos, importação e exportação de dados. É executado como um applet Java, o que permite maior interação com o usuário e o ambiente onde é executado.

Ferramenta: RHT

Descrição: A ferramenta RHT (2006) é muito rápida, estável e de fácil utilização. Funciona com diversos bancos de dados, por utilizar uma camada de abstração. Apesar de não oferecer suporte a plug-ins, possui APIs (Application Programming Interface) que possibilitam a integração com diversos tipos de ferramentas.

Apêndice B

Ferramentas de Automação de Testes Open Source

Ferramenta: TestGen4Web **Descrição:** A ferramenta de testes automatizados TestGen4Web (2006) funciona como uma extensão para o navegador Firefox. Grava as ações do usuário e reproduz dentro do navegador. Permite edição da rotina (script), através de uma interface simples e leve. Inclui um utilitário para exportar as rotinas para diversas linguagens.
Ferramenta: Marathon **Descrição:** Ferramenta de testes automatizados para aplicativos em Java/Swing. Marathon (2006) é um aplicativo estável, com interface clara e simples. Possui interface avançada para edição dos scripts. Suporta gravação e execução de casos de teste e scripts, sem necessidade de programação. Desenvolvido em Java, multi-plataforma, e distribuído sob LGPL.
Ferramenta: Abbot **Descrição:** Abbot (2006) é uma ferramenta de teste funcional para aplicativos Java. Permite criação de rotinas manualmente, ou realizando registro de ações do usuário, através da ferramenta Costello, inclusa no pacote, que também permite controle da execução e da aplicação testada. Muito completo e estável. Por ser uma ferramenta muito utilizada, outras ferramentas livres de teste oferecem integração a esta.
Ferramenta: Selenium **Descrição:** Selenium (2006) é uma poderosa ferramenta de teste funcional automatizado. Permite a captura e registro dos eventos do usuário com uma página Web e posterior reprodução dessa rotina. Interface extremamente simples e de fácil entendimento. Possui recursos avançados de edição das rotinas, além de permitir a criação de rotinas em Java, .NET, Perl, Python e Ruby. É compatível com os principais navegadores Web, e permite executar testes de compatibilidade.

Ferramentas de Testes de Performance Open Source

Ferramenta: Apache JMeter **Descrição:** Apache JMeter pode ser usado para testar a performance de recursos dinâmicos e estáticos, como arquivos, Servlets Java, scripts Pearl, consultas a bases de dados e servidores FTP. Pode simular cargas pesadas de requisições no servidor, na rede ou em uma aplicação, para analisar o desempenho e a estabilidade. Completo e customizável, porém complexo de ser utilizado.
Ferramenta: Open STA **Descrição:** Ferramenta de testes de carga para aplicações HTTP desenvolvido em C++. Interface completa de fácil utilização, que facilita a criação de cenários. Possui uma IDE para desenvolvimento dos scripts, além de diversas opções de monitoração da aplicação sob teste.
Ferramenta: Grinder **Descrição:** Ferramenta de testes de carga, para diversos tipos de aplicação. Utiliza o mecanismo de scripts Jython, que o torna bastante poderoso. Porém, possui uma interface muito limitada, o que dificulta sua utilização, pois é necessário realizar diversas configurações manuais.
Ferramenta: Dieseltest **Descrição:** Ferramenta de testes de carga para aplicações HTTP e HTTPS, desenvolvido em Delphi 5. Interface simples e de fácil utilização. Permite monitorar diversos recursos. Gera gráficos com dados relevantes que facilita a visualização dos resultados.