A IMPORTÂNCIA DOS TESTES DE SOFTWARE NA MELHORIA DA QUALIDADE DOS SISTEMAS COMPUTACIONAIS

João Fukusawa[1]; Orlando Leonardo Berenguel Junior[2]

Resumo
O objetivo deste trabalho é apresentar alguns conceitos referentes às atividades de testes e gestão de qualidade, além de tentar demonstrar como os testes de software podem contribuir para a melhoria da qualidade dos sistemas computacionais. Serão apresentados alguns riscos inerentes às atividades de testes, além dos custos envolvidos em todo este processo. Alguns autores consagrados na área de Engenharia de Software serão referenciados, auxiliando na fixação dos principais conceitos referentes aos testes de software.
Palavras-chave: Engenharia de Software, Teste de Software, Gestão de Qualidade.

Abstract
The objective of this paper is to present some concepts related to testing activities and quality management, in addition to trying to demonstrate how software testing can contribute to improving the quality of computer systems. Will be presented some risks inherent in testing activities, in addition to the costs involved in this process. Some established authors in the field of Software Engineering will be referenced, assisting in the setting of the main concepts related to software testing.
Keywords: Software Engineering, Software Testing, Quality Management.

Introdução

Este trabalho tem como finalidade demonstrar que o emprego da gestão da qualidade e a utilização dos testes de software têm grande importância na melhoria da qualidade dos sistemas computacionais, sendo uma tarefa complexa e que envolve diversos profissionais da área de tecnologia da informação. Dentre as atividades de Verificação e Validação podemos destacar a Inspeção e os Testes de Software. A Inspeção está relacionada à análise de artefatos e documentos, tais como Diagrama de Caso de Uso, Diagrama de Sequências e Código Fonte, sendo assim, considerada uma atividade estática. Os Testes de Software são considerados atividades dinâmicas e estão relacionadas à execução do código para identificação de eventuais erros nos Sistemas de Software. Inicialmente, os Testes de Software eram executados pelo próprio programador. Na medida em que as aplicações tornaram-se mais complexas e devido à evolução da arquitetura dos computadores e dos softwares, os Testes de Sistemas

[1] Graduado em Tecnologia em Processamento de Dados – FATEC – SP. E-mail: joaofukusawa@uol.com.br.
[2] Doutorado em Engenharia de Produção, professor do Instituto Federal de São Paulo Instituto Federal de São Paulo (IFSP) - Campus Bragança Paulista. E-mail: orlandob@ifsp.edu.br.

tornaram-se mais complexos. Além disso, devemos considerar que a integração com as diferentes plataformas que vem surgindo durante os anos também vem contribuindo para esse aumento de complexidade. Houve, portanto, a necessidade de uma maior especialização e atenção às atividades de Validação e Verificação de Software.

Devido ao aumento do investimento em testes de software, algumas empresas criaram novas funções na área de tecnologia da informação, tais como os arquitetos, analistas de testes e testadores, ocasionando a geração de uma mão de obra cada vez mais especializada, prezando pela qualidade e consequentemente, aumentando a satisfação do cliente.

A metodologia a ser adotada por este trabalho é a Pesquisa Bibliográfica por meio de consultas a livros didáticos, revistas especializadas e livros de referência. Outra metodologia a ser empregada é a apresentação de estudos de casos, relatando exemplos de insucesso e sucesso relacionados às atividades de testes de software. Além disto, este trabalho é uma pesquisa de caráter qualitativa fenomenológica e exploratória e o tema abordado é a importância dos testes na melhoria da qualidade dos softwares, destacando a técnica de teste funcional ou comportamental. Serão apresentados estudos de casos referentes ao assunto teste de software, com a finalidade ilustrar como as atividades de testes podem contribuir para a melhoria da qualidade dos Sistemas e como a ausência dos testes pode ser catastrófico.

1 Referencial teórico

Serão apresentados alguns conceitos referentes ao teste de software, conforme alguns autores consagrados na área de Engenharia de Software e a seguir, serão abordados alguns aspectos importantes relacionados à qualidade de software.

1.1 Teste de Software

Algumas definições serão apresentadas abaixo, conforme levantamento bibliográfico em livros didáticos e publicações periódicas.

O Institute of Electrical and Electronics Engineers (IEEE) define os conceitos de defeito, erro e falha, conforme representado pela figura 1, que caso não sejam bem compreendidos, podem causar alguns equívocos entre os profissionais da área de tecnologia da informação envolvidos nas atividades de testes de software.

Figura 1 - Diferença entre defeito, erro e falha, conforme Neto (2007).
Fonte: Neto (2007).

O defeito pode ser entendido como um ato inconsistente realizado por um indivíduo ao tentar compreender uma determinada informação. O erro é uma manifestação concreta de um defeito num artefato de software. A falha é o comportamento operacional do software diferente do esperado pelo usuário (NETO, 2007).

Conforme Sommerville (2011), "o teste tem como objetivo mostrar que um programa faz o que é proposto a fazer e para descobrir os defeitos do programa antes do uso". O autor afirma ainda que o sistema de software deve passar por três estágios: testes de desenvolvimento, testes de release e testes de usuário. Os testes de desenvolvimento são realizados pela equipe de desenvolvimento, podendo ser o programador ou os programadores e testadores trabalhando em pares os responsáveis pelos testes. Tais testes podem ocorrer em três níveis:

(i) teste unitário: ele tem como objetivo testar as unidades individuais de cada programa. O autor defende que os testes unitários, sempre que possível, devem ser automatizados;

(ii) teste de componentes: ele consiste em integrar várias unidades individuais e criar componentes para serem testados; e:

(iii) testes de sistema: ele consiste em integrar os componentes e o sistema é testado como um todo, devendo ser realizado de forma coletiva, podendo ser realizada por uma equipe independente, sem que haja a participação dos projetistas e programadores.

Conforme Sommerville (2011) o teste de release consiste em testar um release de um sistema para uso dos clientes e usuários. Há algumas diferenças a serem consideradas entre o teste de release e o teste de sistema. O teste de release deve ser realizado por uma equipe que não esteve envolvida no desenvolvimento do sistema e tem a função de verificar se o sistema atende aos seus requisitos e é bom para o uso

externo, sendo um processo de teste de caixa-preta, onde o testador se preocupa apenas com as funcionalidades do sistema. Já os testes de sistema realizados pela equipe de desenvolvimento devem se concentrar em achar defeitos. O teste de cenário é uma abordagem de teste de release, na qual o sistema deve ser testado conforme cenários realistas. O cenário é uma situação que descreve uma forma de usar o sistema.

Sommerville (2011) descreve o teste de usuário ou de cliente como um estágio no processo de teste em que os usuários fornecem entradas e conselhos sobre o teste de sistema, sendo essencial pelo fato de que as influências do ambiente de trabalho podem ter um efeito importante sobre a confiabilidade, o desempenho e a robustez de um sistema. Os testes realizados pelo desenvolvedor são inevitavelmente artificiais, sendo difícil representar o ambiente do usuário. Os testes do usuário podem ser classificados em: teste alfa, teste beta e teste de aceitação. O teste alfa caracteriza-se pelo fato de que os usuários do software trabalham com a equipe de desenvolvimento para testar o software no ambiente do desenvolvedor. O teste beta consiste em um release do software disponibilizado aos usuários com o intuito de que eles possam levantar e experimentar os problemas descobertos juntamente com os desenvolvedores. Já no teste de aceitação, os clientes testam um sistema até decidirem se ele está apto ou não para ser implantado no ambiente do cliente. Há seis estágios no processo do teste de aceitação: definir critérios de aceitação, planejar os testes de aceitação, derivar testes de aceitação, executar testes de aceitação, negociar resultados de teste e rejeitar/aceitar o sistema.

De acordo com Neto (2007), a atividade de testes é composta pelos seguintes elementos:

(i) caso de Teste: descrição específica de um teste a ser executado. É composto basicamente por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado;

(ii) procedimento de Teste: descrição das etapas necessárias para a execução dos casos de testes; e:

(iii) critério de testes: selecionar os casos de testes com o intuito de aumentar as possibilidades de provocarem falhas.

Neto (2007) classifica as técnicas de teste de conforme as origens das informações utilizadas para estabelecer os requisitos de teste, contemplando diferentes perspectivas do software. As técnicas existentes são: estrutural e funcional.

A técnica estrutural (ou teste caixa-branca) revela o comportamento interno do software, trabalhando diretamente com o código fonte, avaliando aspectos relacionados com o teste de condição, teste de fluxo de dados, teste de ciclos e testes de caminhos lógicos. Neste tipo de técnica de teste, o código fonte é analisado e são elaborados casos de teste, tentando cobrir a maioria das possibilidades e variações originadas pelas estruturas de condições. (NETO, 2007).

Também chamado de testes de caixa de vidro, os testes de caixa branca possibilitam que o engenheiro de software crie casos de testes que garantam que todos os caminhos independentes foram executados pelo menos uma única vez, além de testar todas as decisões lógicas contemplando os estados verdadeiro e falso. (PRESSMAN, 2011).

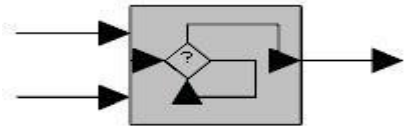

Figura 2 - Técnica de teste estrutural
Fonte: Neto (2007).

A técnica funcional (ou teste de caixa-preta) consiste em testar o software como ele fosse uma caixa-preta, não sendo considerado o comportamento interno do mesmo. O teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. (NETO, 2007).

Chamado também de teste comportamental, possui o foco nos requisitos funcionais do software, não sendo uma alternativa às técnicas de caixa-branca, podendo ser considerado uma abordagem complementar. Os testes de caixa-preta procuram encontrar erros em interface, erros de comportamento ou de desempenho. (PRESSMAN, 2011).

Figura 3 - Técnica de teste funcional
Fonte: Neto (2007).

Bastos et al. (2006) define os responsáveis pela execução dos testes os seguintes profissionais:

(i) líder do projeto de teste: responsável pela liderança do projeto de teste, relacionado a um sistema em desenvolvimento (projeto novo ou manutenção);

(ii) arquiteto de teste: responsável pela infraestrutura de teste, preparando a infraestrutura de teste, montagem do ambiente de teste e escolha das ferramentas de teste;

(iii) analista de teste: responsável pela elaboração dos casos de teste e pelos scripts de teste; e:

(iii) testador: é responsável pela execução dos casos de testes e dos scripts de teste.

Bastos et al. (2006) destaca a importância de outros profissionais na atividade de teste, tais como o usuário (aquele que vai utilizar o software), os desenvolvedores (profissionais envolvidos no desenvolvimento dos sistemas), analistas de produção (profissionais responsáveis por receber o software e verificar se os mesmos estão aptos a entrar em ambiente de produção), gerente de testes, gerentes de qualidade e gerentes de desenvolvimento.

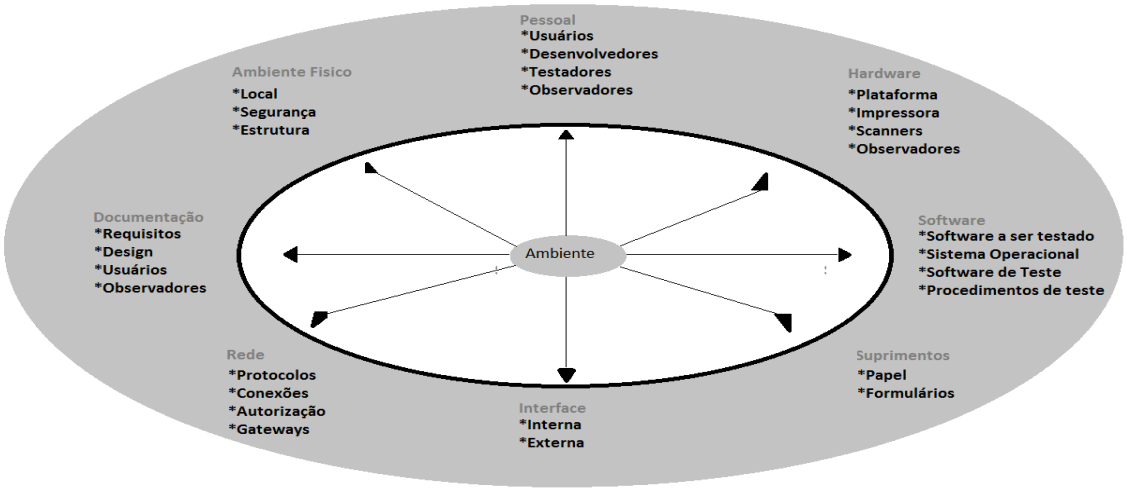

Figura 4 - Ambiente de teste
Fonte: Bastos et al. (2006).

Bastos et al. (2006) destaca a importância da criação de um ambiente de teste, sendo este definido como um ambiente isolado e organizado destinado à descoberta de erros reais (aqueles que realmente ocorreriam se estivéssemos em ambiente de produção). Ao definir um ambiente de teste deverá ser considerado o sistema operacional, a arquitetura do sistema, os componentes, a linguagem de programação e a conectividade entre os ambientes.

1.2 Gestão de Qualidade

De acordo com Pressman (2011) qualidade de software pode ser definida como "uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam".

Considera-se qualidade de software a qualidade do desempenho, dos recursos, a confiabilidade, a conformidade, a durabilidade, a facilidade de manutenção, a estética e a percepção.

Outra visão referente à qualidade de software descrita por Pressman (2011) baseia-se em fatores de qualidade contidos na norma ISO 9126.

O padrão identifica seis atributos de qualidade:

(i) funcionalidade: corresponde ao grau com que o software satisfaz os seguintes subatributos: adequabilidade, exatidão, interoperabilidade, conformidade e segurança;

(ii) confiabilidade: corresponde o tempo que o software fica disponível para o uso conforme indicado pelos seguintes subatributos: maturidade, tolerância a falhas, facilidade de recuperação;

(iii) usabilidade: corresponde ao grau de facilidade de utilização do software conforme os subatributos: operabilidade, facilidade de compreensão e aprendizagem.

(iv) eficiência: corresponde ao grau de otimização do uso, pelo software, dos recursos do sistema, considerando os seguintes atributos: comportamento em relação ao tempo e em relação aos recursos;

(v) facilidade de manutenção: corresponde à facilidade com a qual uma correção pode ser realizada pelo software, sendo indicado pelos seguintes atributos: facilidade de análise, de realização de mudanças, estabilidade e testabilidade; e:

(vi) portabilidade: corresponde a facilidade com a qual um software pode ser transposto de um ambiente para o outro, considerando os seguintes subatributos: adaptabilidade, facilidade de instalação, conformidade e facilidade de substituição.

Segundo Pressman (2011), a garantia da qualidade de software engloba um amplo espectro de atividades concentradas na gestão de qualidade de software, conforme descrito abaixo:

• padrões: Algumas organizações de padronizações tais como o IEEE, a ISO produziu inúmeros padrões para engenharia de software, podendo ser adotados voluntariamente ou impostos pelo cliente ou outros interessados.

• revisões e auditorias: As revisões técnicas correspondem a uma atividade de controle de qualidade que são realizadas por engenheiros de software, com o intuito de revelar erros. As auditorias correspondem a um tipo de revisão realizado pelos integrantes do departamento de Software Quality Assurance (SQA), com o objetivo de assegurar que as diretrizes de qualidade estejam sendo seguidas no trabalho de engenharia de software.

• testes: Corresponde a uma função de controle de qualidade com o objetivo de descobrir erros. O papel do Software Quality Assurance (SQA) é tentar garantir que os testes de software sejam devidamente planejados e conduzidos de forma eficiente.

• coleta e análise de erros / defeitos: A Software Quality Assurance (SQA) reúne e analista os dados de erros e defeitos com o intuito de compreender melhor como os erros são introduzidos e quais tarefas de engenharia de software contribuem para a eliminação dos erros.

• gerenciamento de mudanças: Se as mudanças não forem administradas adequadamente, podem gerar efeitos negativos, prejudicando assim a qualidade. O gerenciamento de mudanças tem um papel primordial para tentar evitar que isso ocorra.

• educação: É fundamental treinar e educar os engenheiros de software, os gerentes e demais interessados no processo de aperfeiçoamento do software.

• gerência dos fornecedores: Qualquer contrato com fornecedores externos deve seguir práticas específicas de qualidade.

• administração da segurança: Toda organização de software deve instituir políticas que protejam os dados em todos os níveis, devendo garantir o emprego de processos e tecnologias apropriadas para que atinja o nível de segurança desejada.

• proteção: O software quase sempre está associado a sistemas que envolvem vidas humanas, podendo os seus defeitos causar resultados catastróficos. Portanto é necessário realizar iniciativas que possam reduzir tais riscos.

• administração de riscos: A análise e a redução de riscos são de responsabilidade dos engenheiros de software. O grupo de Software Quality Assurance (SQA) deve garantir que as atividades de gestão de riscos sejam conduzidas de forma apropriada e que os planos de contingência relacionados aos riscos tenham sido estabelecidos.

2 Pressuposto Teórico

Bastos et al. (2006) relata que durante as décadas de 1970, 1980 e 1990, os testes de software eram realizados pelos próprios desenvolvedores de software, contemplando

o que atualmente conhecemos por testes unitários e testes de integração. Com o advento dos sistemas para Internet, do aumento da complexidade das aplicações e do surgimento das novas tecnologias, fez com que as empresas procurassem melhorar a qualidade dos softwares a serem desenvolvidos, havendo uma maior especialização dos profissionais envolvidos na atividade de testes.

Para a realização dos testes de softwares é necessário um conhecimento das técnicas existentes (estrutural e funcional), dos estágios (testes de desenvolvimento, testes de release e testes de usuário), das atividades e dos profissionais envolvidos nos testes dos sistemas.

Segundo Pressman (2011), o software é testado com o intuito de revelar erros cometidos durante o projeto do software, devendo ser realizado pelos especialistas de testes através de uma estratégia para testes elaborada pelos gerentes de projeto, engenheiros de software e pela equipe de testes.

Segundo Bastos et al. (2006) as atividades de controle de qualidade se baseiam na identificação de defeitos. Essas atividades devem ocorrer desde o início do processo de desenvolvimento do software até o teste final do software e a implantação do sistema. Uma metodologia de qualidade muito utilizada é a FURPS *(Functionality, Usability, Reliability, Performance, Supportability).*

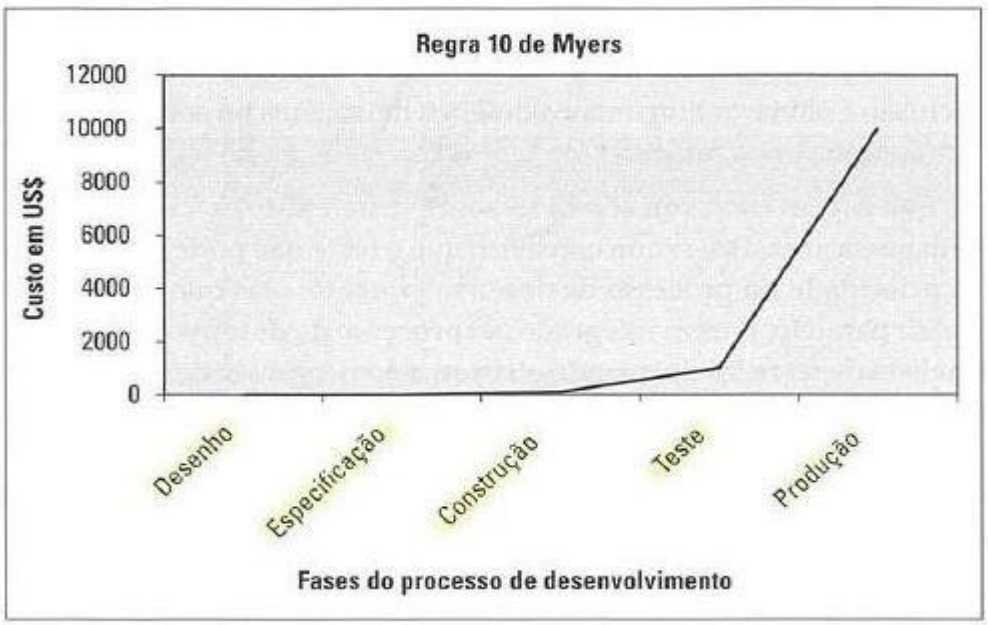

Figura 5 - Custo da correção dos defeitos
Fonte: Bastos et al. (2006)

Bastos et al. (2006) descreve que o custo de correção dos defeitos possui a tendência de aumentar quanto mais tarde o defeito for detectado, conforme a Regra 10

de Myers reproduzida na figura 5. Os defeitos encontrados durante a produção possuem a tendência de custar muito mais do que defeitos encontrados em outros documentos do projeto do sistema.

Os profissionais responsáveis pelos testes de software devem começar as suas atividades no início do desenvolvimento do sistema, com o intuito de detectar defeitos nas fases iniciais do projeto. Inicialmente o custo será maior devido ao investimento necessário para montar a infraestrutura física e a alocação de profissionais. Por outro lado, haverá uma redução no custo da correção dos defeitos.

Conforme Bastos et al. (2006), a atividade de testar está relacionada ao risco, e que quanto maior a cobertura dos testes, maiores serão os investimentos para tentar garantir que nenhum defeito seja encontrado quando o software estiver sendo utilizado pelo usuário em ambiente de produção. A maior parte das empresas que possuem equipes de teste devem procurar um nível que minimize a possibilidade de defeitos graves, pois os prazos devem ser cumpridos. O risco é uma variável importante a ser levado em consideração no momento de ser elaborar o projeto de teste de uma aplicação.

É de grande importância realizar uma análise de riscos, considerando os seguintes aspectos:

- A probabilidade de ocorrência do risco.
- O impacto e a perda associados a esse risco.
- Bastos et al. (2006) destaca alguns dos possíveis riscos associados ao processo de testes:
- Orçamento: Limitação no orçamento para execução completa das atividades de teste poderá prejudicar a abrangência dos testes.
- Qualificação da equipe técnica de teste: A equipe deverá estar preparada para testar o software.
- Ambiente de teste: O ambiente de testes deverá estar semelhante ao ambiente de produção, inclusive em relação às ferramentas.
- Ferramentas: Em alguns casos em que for necessária a realização de testes automatizada, a falta de uma ferramenta poderá acarretar atrasos no cronograma.
- Metodologias: Deverá ser utilizada uma metodologia adequada para a realização dos testes.

- Cronograma: No caso em que os prazos sejam muito exíguos, deverão ser priorizados os programas cuja incidência de defeitos represente um maior risco para o negócio.
- Novas tecnologias: A falta de conhecimento de uma nova tecnologia por parte da equipe de testes pode ser um fator de risco importante, devendo ser contornado com um treinamento adequado.

Pfleeger (2007) descreve em sua obra alguns estudos de casos relacionados à atividade de teste, na tentativa de ilustrar a importância que os testes possuem na melhoria da qualidade dos sistemas de software.

Entre junho de 1985 e janeiro de 1987, uma máquina de terapia de radiação, denominada como Therac-25, foi envolvida em seis acidentes sérios, causando a morte e problemas de saúde devido à overdose de radiação em diversas pessoas. Neste caso o software foi construído por uma única pessoa, que reutilizou o código de uma máquina anterior. Uma parte do software foi testada em um simulador, mas a maior parte foi testada como parte de um sistema maior, utilizando-se de testes de sistemas integrados. Neste caso houve poucos testes de unidade e de software. Foram realizadas análises por empresas externas e ficou comprovado que deveriam ser realizados testes de exaustão.

Uma notícia divulgada em agosto de 1997 a respeito de que um possível erro de software tenha prejudicado o sistema de radar do aeroporto de Guam, e tenha causado a queda do jato da Korean Air, matando 225 pessoas, causou um grande desconforto na imprensa. Na ocasião o jato estava voando muito baixo. O sistema chamado de Radar FAA alerta a altitude de segurança mínima, enviando um aviso aos oficiais que estão no solo. Mas isso não ocorreu. O sistema não enviou o alerta naquele dia. Foi verificado que o software foi alterado para que o sistema deixasse de emitir tantos alarmes falsos, porém a modificação realizada não estava correta, pois considerou como falsa um alerta verdadeiro, causando assim o acidente com o jato da Korean Air.

Mas não é só de casos de insucesso que norteiam os testes de software. Algumas empresas como a Motorola, utilizam modelos que procuram a melhoria da qualidade nas atividades de teste. A empresa faz o uso do modelo chamado teste de ausência de falhas, que consiste em uma derivação de uma função da freqüência de falhas.

Pfleeger (2007) destaca o exemplo da International Business Machines (IBM), que propuseram um novo processo de desenvolvimento de software denominado *cleanroom***.** A abordagem deste novo método envolve dois princípios fundamentais:

- Certificar o software durante as especificações ao invés de esperar pelos testes de unidade para a detecção de eventuais erros.
- Tentar produzir softwares sem defeitos.

Um dos princípios básicos do *cleanroom* é a medição estatística dos testes.

3 Conclusões

Conforme relatamos neste trabalho, as atividades de testes vêm se destacando no cenário da Engenharia de Software, devido ao aumento da complexidade dos sistemas computacionais e ao maior grau de exigência do usuário final. A gestão de qualidade tornou-se algo fundamental para que os testes sejam realizados com maior eficiência e eficácia.

Anteriormente, a Qualidade era considerada um diferencial que as Empresas ofereciam aos seus Clientes. Atualmente, tornou-se um pré-requisito para a sua sobrevivência, devido a um mercado cada vez mais competitivo e qualificado.

A princípio pode parecer árdua a tarefa de se testar um Software, porém não podemos nos esquecer que infelizmente, os erros fazem parte do cotidiano dos desenvolvedores e devemos aprender com os erros para que eles não se repitam, buscando sempre a melhoria dos resultados, independente da metodologia de desenvolvimento de software a ser utilizada.

Os conceitos referentes às atividades de teste e à gestão de qualidade apresentados, certamente irão contribuir para a melhoria da qualidade dos sistemas computacionais, na medida em que proporcionarão uma maior consciência de que os sistemas de software devem ser construídos de forma íntegra, confiável e eficiente.

Referências

BASTOS, A. et al. **Base de Conhecimento Em Teste de Software**. 2ª., Ed. São Paulo: Martins Fontes, 2007.

FACHIN, Odília. **Fundamentos de metodologia**. 4ª. Ed. São Paulo: Saraiva, 2003.

NETO,Arilo. Introdução a Teste de Software. **Revista Engenharia de Software**, Rio de Janeiro,v. 1, n.1, 2007. Disponível em: < http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035>. Acesso em 05 mai. 2013

PFLEEGER, Shari L. **Engenharia de Software**, 2ª. Ed. São Paulo:Pearson Prentice Hall, 2007.

PRESSMAN, Roger S. **Engenharia de Software:** Uma abordagem profissional. 7ª. Ed. São Paulo: McGraw Hill, 2011.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª. Ed. São Paulo: Pearson Prentice Hall, 2011.